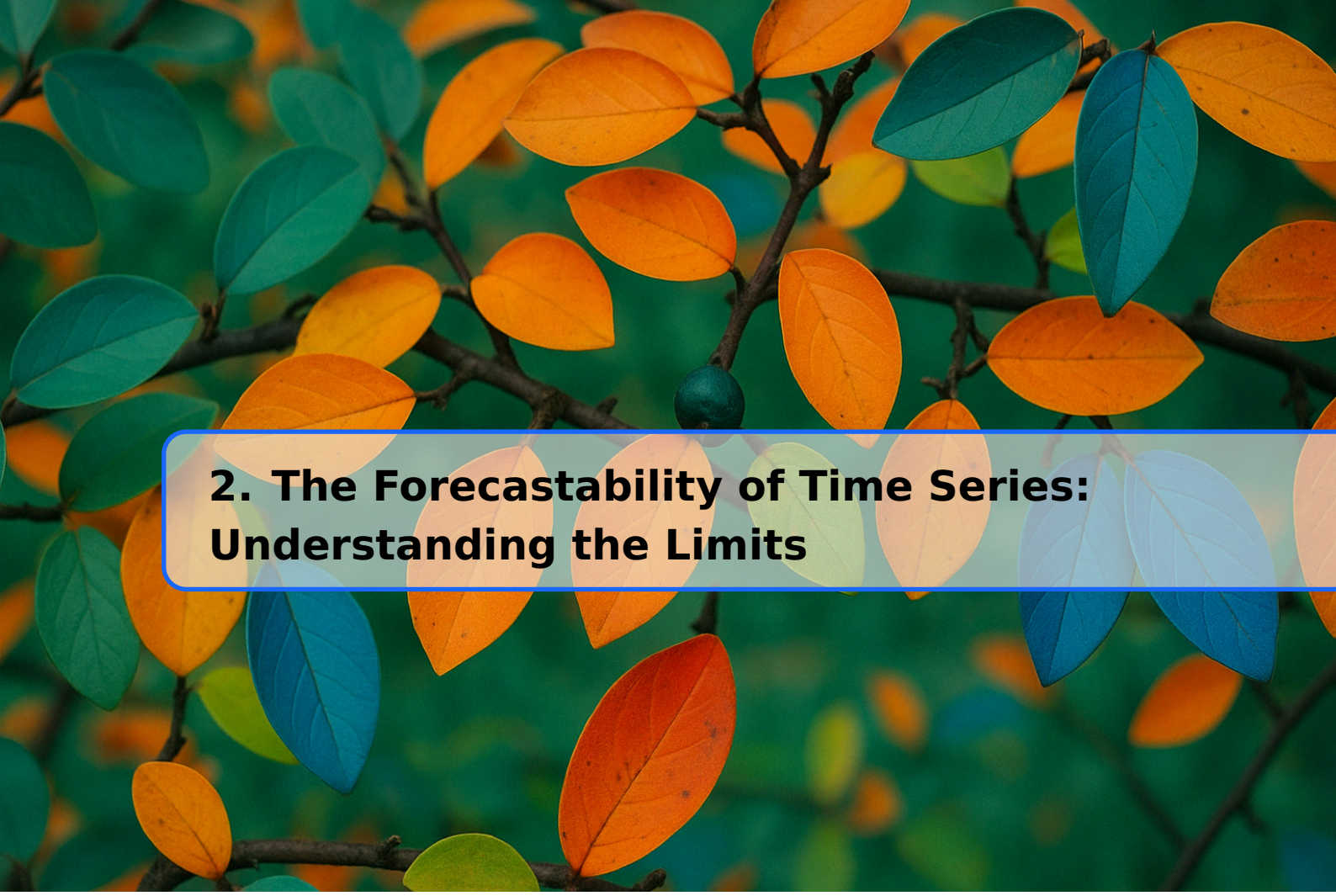

2. The Forecastability of Time Series: Understanding the Limits

> *”Before we talk about models, let’s ask: is this series even forecastable?. In other words, before jumping to build a model, we should ask: is there enough signal to forecast at all?”*
>
> **— *Forecasting Proverb***

Chapter 1 traced the evolution of forecasting from ancient civilizations to modern data-driven methods, revealing how the quest to anticipate the future has advanced from symbolic divination to sophisticated machine learning modeling. It outlined the foundational principles of time series analysis, explored early mathematical contributions from figures like Yule, Slutsky, and Wold, and highlighted how stochastic processes, autoregression, and decomposition theories reshaped our understanding of dynamic systems. The chapter also reviewed the institutionalization of forecasting, from ARIMA models and state-space filters to the emergence of machine learning and hybrid techniques in the 21st century. Finally, it emphasized that accurate forecasting is not just about model selection but about recognizing the nature and limits of predictability embedded in the data itself. With this historical and conceptual foundation, we now turn to the building blocks of practical forecasting, starting with how to formally diagnose, prepare, and model time-series data.

Before embarking on any time series forecasting endeavor, a crucial yet often overlooked question must be addressed: Is the series we are analyzing actually forecastable?

This chapter explores the fundamental nature of forecasting, examining when forecasting methods are appropriate and when alternative approaches should be considered. We investigate the philosophical and practical limits of predictability, distinguishing genuine forecasting problems from other predictive tasks such as classification, anomaly detection, and cross-sectional regression.

Time series vary dramatically in their inherent predictability, from highly predictable patterns with clear seasonal cycles and stable trends to nearly random processes that defy meaningful prediction. Understanding these limitations is not merely academic; it has profound practical implications for businesses and researchers. By learning to quantify forecastability and recognize when forecasting is likely to succeed or fail, forecasters can avoid wasted effort, inappropriate models, and misleading predictions. This chapter provides the conceptual foundation and practical tools needed to assess when forecasting is the right approach and what level of accuracy can reasonably be expected, enabling more informed decision-making in the face of an uncertain future.

2.1 The Nature of Forecasting

Forecasting is commonly defined as the process of making informed statements or predictions about future events or conditions that are not yet known. It is fundamentally about anticipating what has yet to happen, typically by analyzing existing data, identifying patterns or trends, and extrapolating them forward. In practice, forecasting is often described as an art and a science, combining quantitative analysis with expert intuition to predict future outcomes. For example, operations research and business scholars note that a forecast may involve statistical models (e.g. time-series extrapolation or causal models) adjusted by managerial judgment. In essence, forecasters use the best available information from the present and past to make rational estimates about the future, recognizing that these estimates are provisional and subject to error. A key element in the nature of forecasting is uncertainty. Forecasts are only needed when there is uncertainty about what will happen. If an outcome is predetermined or fully controllable, there is no need to forecast it. Indeed, Rob J. Hyndman, a leading scholar of forecasting, defines the discipline as a probabilistic endeavor focused on predicting future values of a time series while explicitly quantifying uncertainty. In his seminal work, "Forecasting: Principles and Practice" [48], he emphasizes: "Forecasting is not about eliminating uncertainty, but understanding it and communicating it through prediction intervals." In statistical terms, this often means providing prediction

intervals or ranges of possible outcomes, not just single-valued predictions. The economist Frank H. Knight famously distinguished measurable risk from true uncertainty, essentially defining uncertainty as our 'inability to forecast the likelihood of events happening in the future' [61]. This highlights the philosophical reality that not all aspects of the future are knowable – some future events (for example, unexpected innovations, crises, or 'black swan' events) lie beyond the realm of calculable probability. No matter how much data is collected, there is always residual uncertainty about the future.

Because of these limits, scholars have long pointed out the need for epistemological humility in forecasting. As early as the 20th century, critics noted that forecasting can be notoriously difficult, especially in social and economic domains, leading economist John Kenneth Galbraith to quip that the only function of economic forecasting is to make astrology look respectable [33]. Although meant humorously, Galbraith's remark underscores a real point: forecasts often have wide margins of error, and even expert forecasters cannot predict the main turning points. The occurrence of unexpected shocks (a dramatic example being the COVID-19 pandemic) illustrates why even the best models should be viewed as indicative rather than infallible. In other words, a forecast is a reasoned judgment about the future, not a definitive claim of truth. To manage this, forecasters typically update their predictions as new information arises and design scenarios or contingency plans to cope with the possibility that reality may diverge from their central forecast.

Furthermore, science philosophy offers insight into the predictive limits of our knowledge. Explanation is not the same as prediction – a scientific theory might explain why certain events occur, yet still fail to predict the next occurrence of those events. As an example, Joshua Epstein argues that a model can greatly enhance understanding of a phenomenon without necessarily producing accurate forecasts; he notes that plate tectonic theory explains why earthquakes occur even though it cannot predict the exact time and location of the next earthquake [30].

Complex systems, such as economies, ecosystems, or climate, often exhibit nonlinear dynamics and intricate feedback loops, making precise long-term forecasting exceptionally difficult. One of the key reasons for this unpredictability lies in their sensitivity to initial conditions. Even a minuscule variation at the beginning can lead to drastically different outcomes over time. This concept is famously encapsulated by the "butterfly effect," a term coined by the meteorologist Edward Lorenz. While studying weather models, Lorenz discovered that tiny differences in initial data, such as rounding off a number, could result in entirely different weather predictions. The metaphor he used was that the flap of a butterfly's wings in Brazil could theoretically trigger a tornado in Texas weeks later. This vividly illustrates how small, seemingly insignificant inputs can cascade into massive, unpredictable changes in complex systems.

Lorenz's meteorological work revealed that there is a horizon of about two weeks beyond which detailed weather forecasts become unreliable due to the chaotic nature of atmospheric dynamics [65]. This finding, now widely accepted in the weather and climate community, exemplifies an inherent limit of predictability in certain systems: beyond a certain point, the future state of the system diverges so much that no forecast can be confident. Forecasters thus must grapple not only with statistical uncertainty, but with fundamental limits of predictability imposed by the nature of the system itself.

Different academic disciplines bring additional perspectives to what forecasting entails and how it should be approached. Climate science provides a clear cross-disciplinary

view on forecasting under deep uncertainty. Climate scientists distinguish between a forecast/prediction and a projection to emphasize the role of assumptions and scenarios. A projection is defined as a 'potential future evolution' of a system, often calculated using simulation models, and is explicitly conditional on assumptions about factors such as future emissions or policies.

In contrast, a prediction would imply a more definite expectation. In long-range climate forecasting (such as projecting global temperatures by 2100), scientists present multiple scenario-based projections rather than a single deterministic forecast, precisely because the actual outcome will depend on uncertain human choices and natural variability. This practice reflects a broader epistemological stance: instead of claiming certainty, climate forecasters acknowledge the limits of knowledge by exploring a range of plausible futures and attaching confidence levels or probabilities to them. Even in shorter-term environmental forecasts (such as seasonal rainfall or river flows), language and methodology often highlight uncertainty — for example, hydrologists might provide probability distributions for flood occurrence rather than a yes/no prediction, recognizing that many variables can change. The climate science approach thus enriches the nature of forecasting with the concepts of scenarios, ensemble predictions, and uncertainty quantification as core components.

These developments illustrate the growing interest in data-driven forecasting, particularly through machine learning. However, they do not change the fundamental reality: When a time series lacks structure, no model, regardless of complexity, can reliably predict its future.

This reinforces the distinction between forecasting tools and forecastability itself. Advanced models can only succeed when the underlying data exhibit temporal patterns, signal regularity, or memory. When time series are dominated by noise or contain few exploitable dynamics, even the most sophisticated methods will fail.

Later chapters in this book will explore various forecasting models, from statistical to machine learning to deep learning. Here, our focus remains on the foundational question: *is there anything to forecast at all?*

In general, a rich understanding of the nature of forecasting emerges from these varied points of view. Forecasting is not simply about guessing the future; it is a disciplined process of gathering evidence, quantifying uncertainty, and making reasoned projections in the face of limited knowledge. It operates at the intersection of data and uncertainty, requiring both analytical tools and thoughtful judgment. Philosophically, it confronts profound questions about what can be known and how confidently we can extend our knowledge beyond the present. Practically, it has become an indispensable activity in nearly every field, from scientists predicting climate change, engineers planning infrastructure, to business leaders projecting sales, each incorporating the principles of uncertainty, model limitations, and continuous learning. By integrating statistical models with an awareness of uncertainty and drawing on cross-disciplinary insights (from climate science scenarios to AI pattern recognition), forecasters aim to make the best possible statements about the future.

The nature of forecasting, therefore, is one of blending knowledge and humility. It is the art of making the uncertain future a little more knowable while never losing sight of the uncertainties that remain at its core.

2.1.1 When Is a Problem a Forecasting Problem (and When Is It Not)?

Forecasting generally refers to predicting future values or events based on historical and current data, typically in a time-ordered sequence. In contrast to a random guess, a forecast is based on data and a logical analysis of trends. For example, forecasting might involve estimating the sales of the next quarter by analyzing past sales trends, seasonality, and other temporal patterns. A concise definition by Hyndman & Athanasopoulos [48] is that forecasting is about predicting the future as accurately as possible, given all available information, including historical data and knowledge of future events that could affect forecasts. In essence, forecasting is a specialized form of predictive modeling that explicitly deals with time-indexed sequential data and seeks to extrapolate observed patterns into the future.

The key assumptions underlying forecasting are that historical patterns and relationships will persist (at least approximately) into the future. Time-series forecasting methods work by identifying components such as trend (long-term increase / decrease), seasonality (repeating cycles), and autocorrelations in the data, under the assumption that these can be projected forward. Time-series forecasting assumes that historical patterns will continue in the future and uses statistical techniques to extrapolate those patterns. This assumption is what enables forecasts, but it is also a potential weakness: If a system undergoes a major change or regime change, the old patterns may no longer hold. Sudden changes in conditions or unforeseen events can alter historical trends and degrade the accuracy of the forecast. In other words, forecasting works best in relatively stable environments or in places where changes are gradual and can be anticipated.

Most statistical forecasting methods rely on the concept of stationarity, the idea that the underlying statistical properties of the series (mean, variance) do not change over time. A strictly stationary time series has a constant distribution over time. In practice, many time series are not stationary (they have trends, seasonal effects, or evolving variances), but forecasters often transform or model these components to achieve stationarity in the residuals. Stationarity is important because many forecasting models theoretically require a stationary process to make reliable predictions.

If one naively regresses one non-stationary time series on another, it can lead to spurious results that appear to fit well in the sample but fail to predict future points. Thus, a forecaster will check for stationarity and apply techniques such as differencing or detrending to stabilize the series before modeling. The takeaway is that forecasting assumes a degree of continuity and stability in the data-generating process (after suitable modeling of trends/seasonality), so that the future can be viewed as a logical extension of the past.

Another assumption (or prerequisite) for forecasting is the availability of sufficient historical data. Without enough past observations, it is difficult to detect meaningful patterns or train a predictive model. In cases where no historical data exist (e.g. a completely new product or an unprecedented situation), statistical forecasting methods cannot be directly applied; one must resort to judgmental estimates or analogies. For example, Hyndman [48] notes that when faced with a complete lack of historical data or a uniquely new market condition, judgmental forecasting (expert opinion, scenario analysis) may be the only option. Although judgmental methods produce a 'forecast', the lack of data-driven pattern means that the problem is not a traditional forecasting exercise in the modeling sense – it becomes more of an expert prediction problem.

In summary, forecasting is characterized by time-ordered data, an aim to predict future time

points, and an assumption that underlying patterns (trend, seasonality, correlations) will continue into the forecast horizon (absent external disruptions).

2.1.1.1 Criteria for When Forecasting is Appropriate

Not every prediction problem is a forecasting problem. Certain criteria should be met for forecasting methods to be appropriate:

- **Sequential, Time-Indexed Data:** The data should be recorded over time with a meaningful chronological order (e.g., daily stock prices, monthly sales, yearly population). Forecasting is inherently temporal; the order of observations matters, and we care about predicting at future time steps. If shuffling the data would destroy its meaning, that is a sign the sequence is important. In forecasting problems, the context of each observation is its position in time relative to others.
- **Presence of Temporal Structure:** There should be evidence of temporal patterns or dependencies that can be exploited for prediction. This could include trends (long-term increases or decreases), seasonal cycles (regular patterns such as weekly sales spikes or annual climate effects), or autocorrelation (where past values influence future values). If the series shows such a structure (for example, a strong seasonal pattern in sales each year), forecasting models can leverage it to project forward. Forecasting methods such as time series decomposition, exponential smoothing, or ARIMA explicitly model these patterns. Without any temporal structure (if the data were pure noise), the best a forecast can do is predict the long-term mean or the last observed value, which yields little informational gain. In short, forecasting is appropriate when past behavior of the time series is a reasonable guide to its future behavior
- **Stationarity (after Accounting for Patterns):** As noted above, many forecasting techniques assume stationarity or work better under stationarity. This does not mean that the raw series must be stationary (a few real-world series are), but it means that we can transform or model the series in a way that the residuals are stationary. For example, a non-stationary series with a trend can often be differentiated to achieve stationarity. Forecasting is most reliable when the statistical properties of the process are stable over time (or have been made stable through modeling). When a series is (at least approximately) stationary, the relationships estimated from historical data can be expected to hold in future periods. If you identify that the series is prone to drastic, non-repeating changes, the use of standard forecasting models becomes questionable.
- **Sufficient Historical Data and Seasonality Coverage:** There should be enough historical observations to detect the patterns mentioned above. The required amount varies by context, but as a rule, you would like multiple cycles of any seasonal period (e.g. 2-3 years of monthly data to capture annual seasonality) and enough data points to reliably estimate model parameters. Forecasting methods struggle with extremely short series because one cannot distinguish signal from noise or seasonality from one-offs with just a few points. The more data (and the more repetitions of patterns) available, the more justifiable a forecasting approach becomes.
- **Forecast Horizon within Reasonable Bounds:** Forecasting is suitable when the goal is to predict future values for a given horizon (short-term, medium-term, long-term) and that horizon is such that historical data is still informative for it. Generally, the farther out one tries to forecast, the more uncertainty and the less one can rely on past patterns. If the required horizon is so far beyond the historical data (e.g. forecasting 50 years ahead based on 5 years of data), the problem might be ill-posed for standard

forecasting (and might require scenario planning instead). However, deciding the horizon is part of formulating a forecasting problem, and most forecasting methods allow various horizons (with diminishing accuracy as it extends).

- **Domain Knowledge Supports Forecasting:** Often, an understanding of the domain can confirm that forecasting is appropriate. For example, in energy demand forecasting, we know that there are daily and weekly cycles and trends due to population growth; thus, using past demand to forecast future demand is sensible. If domain knowledge suggests that the system is largely driven by random external shocks or one-time events, a pure forecasting approach (which extrapolates patterns) may not be the right tool.

In practice, if a dataset and question meet these criteria, time-indexed data with temporal dependencies, a stable or modeled-as-stable process, and a need to predict future values, then it qualifies as a forecasting problem. The forecaster would then choose a time series model (such as ARIMA, exponential smoothing, or a machine learning model designed for sequence data) and generate forecasts with uncertainty estimates. Many textbooks emphasize identifying the structure of the time series (trend, seasonality, etc.) as a first step in deciding that a forecasting model can be applied. These assumptions distinguish forecasting from other types of predictive tasks. When these assumptions hold reasonably well, we have the necessary conditions for treating a problem as a forecasting problem.

If these conditions are not met, the problem may not qualify as a forecasting problem. Let us illustrate these points with industry-specific examples.

- **Finance**: In finance, many asset price series lack a strong temporal structure, making forecasting challenging. A prime example is stock market returns, which often resemble a random walk: Past price changes do not reliably predict future changes. According to the Efficient Market Hypothesis, new information is quickly absorbed into prices, so the price movement of tomorrow is essentially independent of the movement of today[70]. This implies little to no autocorrelation in stock returns, meaning short-term price forecasts have no skill beyond coin-flip accuracy. However, not all financial time series are devoid of predictable structure. Volatility (the variability of returns) famously exhibits autocorrelation and mean reversion — periods of high volatility tend to be followed by a return to more normal volatility. This volatility clustering means that risk metrics can be predicted with some precision. In fact, ARCH / GARCH models (Engle [28]; Bollerslev [10]) leverage this temporal dependence in variance to predict future volatility. For example, a GARCH model will predict tomorrow's volatility to be high if today's returns were highly volatile, capturing the persistence of risk. Credit risk metrics also show temporal structure: default rates and credit spreads usually exhibit serial correlation (a high default rate this year often implies a higher than average default rate next year) [91]. These credit risk variables often mean-revert over business cycles, so models can be built to forecast their future values as economic conditions revert to the norm. In summary, financial variables that behave like random walks (e.g., asset prices in an efficient market) violate the 'temporal structure' criterion and are hard to forecast, while those with mean-reverting or cyclical patterns (volatility, credit risk indicators) meet the criteria and can be forecast with some success using time series models [29].
- **Energy**: The energy sector provides textbook cases of series rich in temporal structure and thus amenable to forecasting. Electricity load (power demand) is a prime example: it shows clear daily cycles (high demand during day, lower at night), weekly patterns

(weekday vs. weekend usage), and seasonal variation (e.g. peaks in summer or winter due to cooling/heating) superimposed on any long-term trend. Such multiple seasonalities and trends constitute strong temporal structure that forecasting models can exploit. In fact, electric load often involves two prominent seasonal periods – a 24-hour daily cycle and an annual cycle – that can be explicitly modeled. Because these patterns repeat with some regularity, forecasting techniques (from classical time series models to machine learning) excel at predicting future electricity demand based on past observations. Empirical studies and global competitions have confirmed the forecastability of energy loads. For instance, participants in the Global Energy Forecasting Competitions [45] achieve high accuracy by incorporating calendar effects and weather variables that drive predictable demand swings. Research by Hyndman [31], Hong [47], and others [27] consistently shows that methods capturing seasonality and temperature-dependence (e.g. weather-based regression, exponential smoothing with multiple seasonal periods, neural networks with calendar features) yield very accurate load forecasts. In short, the electric power domain meets the key criteria (stationary seasonal patterns and dependable relationships with exogenous factors like temperature), making it one where forecasting is highly appropriate and effective.

2.1.1.2 When Forecasting May Not Be Appropriate

There are many predictive problems for which forecasting methods (as defined above) are not the right approach — in such cases, other approaches (classification, anomaly detection, etc.) should be considered instead. Key situations where a problem would not be considered a forecasting problem include the following:

- **No Meaningful Temporal Order:** If the data are not a time series or sequence, or if the order of observations can be permuted without affecting the analysis, then it's not a forecasting scenario. For example, predicting whether an email is spam based on its content is a classification problem – there is no time component involved in the prediction (even though emails arrive over time, the prediction does not use temporal patterns). Similarly, if you have a dataset of customer attributes to predict who will churn, you might use the time of observation as just another feature, but you are not forecasting a time-dependent sequence for each customer; you are classifying customers into "will churn" or "will not churn." Similarly, if one tries to predict an outcome that doesn't evolve over time (like a static physical constant or a one-off event), time-series forecasting is not. In such cases, applying time series forecasting techniques would be inappropriate. The absence of sequential structure means we should use other predictive modeling methods (classification, static regression, etc.) instead of forecasting.
- **Predicting Categories or Classes (Classification Tasks):** If the outcome of interest is a categorical label or class (e.g. will a machine fail or not, which product a customer will buy, what category an observation falls into), this usually is not a forecasting problem, but a classification problem. Forecasting typically produces a numeric or continuous output (or occasionally a continuous probability or count) for future time points, rather than a discrete class label. For instance, forecasting might estimate the value of a stock price tomorrow, whereas a classification would predict whether the stock will go up or down (two classes) – the latter is not a forecasting problem per se, but rather a classification or binary prediction problem, even though it is about the future. Forecasting typically produces a numeric time-indexed prediction (e.g. next quarter's sales), whereas predicting a class label (e.g. bull vs bear market) is

a classification task, even if it concerns the future. Therefore, a question like 'Will stock X go up or down tomorrow?' is better treated with classification methods, not time series forecasting. Likewise, forecasting in a call center context might predict the number of calls next hour, while an event prediction might classify the hour as "busy (yes/no)" based on a threshold. If the core question is framed as identifying a class/category, then methods like logistic regression or decision trees (classification algorithms) are appropriate, not time series forecasts.

- **Lack of Temporal Pattern or Signal:** If the time series data is essentially random or has no discernible pattern, forecasting becomes trivial or meaningless. For example, a purely random sequence (when formal tests would deem it 'white noise') has no structure to extrapolate; the best forecast is essentially the average (or last observed value) with very large uncertainty. One cannot meaningfully improve on that with fancy models. In such cases, we might say the problem doesn't qualify as forecasting in a useful sense, because there is no pattern to forecast. Another example: if data are dominated by high volatility or wild fluctuations that are not predictable (sometimes termed a chaotic or near-random process), then a forecasting model will have little utility beyond perhaps predicting an overall range. Without a stable pattern, treating it as a forecasting problem might lead to large errors. Financial return series often appear nearly random – the best forecast may just be 'no change'. In practice, analysts might then shift focus to nowcasting (predicting the present from partial data) or simply reporting descriptive stats rather than forecasting far ahead.
- **Data Dominated by Anomalies or Regime Changes:** When the historical data is highly irregular due to anomalies (outliers, one-off events) or if there has been a recent regime shift, forecasting methods that rely on historical patterns may be unsuitable. For instance, consider a time series of website traffic where most days follow a predictable pattern, but occasionally a viral event causes a huge spike. A forecasting model might be thrown off by those spikes (as they violate the regular pattern) – forecasting in such a case might either under-predict spikes or over-predict normal days if it tries to account for the spike. If anomalies are frequent or dominate the series, one might focus on anomaly detection methods instead of forecasting future values. Additionally, if a structural break has occurred – say a new policy or technology fundamentally changed the process last month – then using data from before that break to forecast after it can be very misleading. Forecasting assumes continuity; a regime change breaks that continuity. In those scenarios, it might be better to handle the segments separately or use scenario analysis instead of naive forecasting. As an example, many economic forecasts failed at the beginning of the COVID-19 pandemic because models based on years of stable economic growth could not account for the abrupt regime shift; new approaches or human judgment had to be introduced. Time series methods are sensitive to outliers and sudden changes, and forecasts 'may not be accurate if significant changes occur in the business environment'. When such conditions apply, one should be skeptical of forecasting; the problem might need reformulation or alternative techniques.
- **Primarily Cross-Sectional or Static Problems:** If the main predictive problem involves understanding the relationships between variables at a point in time (cross-sectional data) rather than predicting over time, it is not a forecasting problem. For example, predicting someone's credit score from their income and debt is a regression problem in cross-sectional data, not a time series forecast — although it's a prediction, it's not about future time-indexed data. Similarly, image recognition (such as classifying photos) or regression tasks such as predicting the price of a house from

features are not forecasting problems because they do not involve extrapolating a time series. In such cases, trying to force a forecasting approach (which expects sequential data) would be inappropriate.

- **Event Prediction vs. Time Series Forecast:** A special case is predicting the occurrence of specific events (which might be one-time or irregular) – for example, predicting when a machine will break down or whether a customer will default on a loan. If the goal is to estimate time-to-event or the probability of an event by a certain date, one might use survival analysis or probabilistic classification, rather than standard time series forecasting. Forecasting could be used if one converts the problem (e.g. forecasting the number of failures in each future time interval), but predicting a one-off event occurrence is usually not formulated as a time series forecasting task. It's more of a classification or hazard modeling task. Thus, if your prediction target is an event occurrence (yes/no or time until event) rather than a value at each time interval, you are generally outside the realm of classical forecasting. In other words, if time order is irrelevant to the prediction, then traditional regression or classification approaches suffice.

In essence, a problem does not qualify as forecasting if it lacks a time-indexed quantitative target or if the primary aim is something other than predicting a future value of a series. As a simple rule of thumb: if you can ask the question 'What will the value / amount of X be at time t + h?' (for some future horizon h), it is a forecasting problem. If the question is instead 'Which category does X belong to?' or 'Is X abnormal?' or 'What is the relationship between Y and Z?' without a future-time element, it is not a forecasting problem. Instead, it could be a classification, anomaly detection, or regression (non-temporal) problem. Even within time-series analysis, there are tasks like clustering or classification of time series, but these are distinct from forecasting. For example, clustering time series groups similar historical patterns; classification of time series might assign labels to entire sequences (like identifying which category a given time series belongs to); these tasks analyze time series data, but do not produce forecasts of future values.

Finally, it is worth noting that sometimes a predictive task can be approached in multiple ways. For instance, one could predict an electricity outage by either forecasting the load on the grid (a time series of load and flag when it exceeds a threshold) or by directly classifying days as outage vs. normal based on predictors. If one chooses the latter route (direct classification), then one is not using a forecasting approach. The right formulation depends on the nature of the data and the question. If temporal dynamics are complex and informative, a forecasting formulation is warranted; if not, a static predictive model might suffice.

2.1.2 Forecasting vs. Other Predictive Modeling Approaches

Classification involves predicting a discrete class label for each example, rather than a numeric value over time. In classification, the order of observations is usually irrelevant; what matters is the features of each instance. A classic example is classifying images or emails – each instance is independent and has a label. Time can be a feature in classification (for example, fraud detection might use time of transaction as a feature), but the goal is not to predict a future numeric sequence, it is to assign a category.

Forecasting is a technique that takes data and predicts the future value of the data by looking at their unique trends. Forecasting factors in a variety of inputs predicts future behavior, not just a number. In forecasting, there is typically no separate output variable apart from the

time series itself – we use the past values of a series (and possibly time-related exogenous variables) to predict its future values. In classification, we often have several input features to predict a separate target variable. For example, forecasting might tell us the expected total employee turnover rate next year (a numeric value %) based on past rates. In contrast, a classification approach might identify which specific employees are likely to quit in the next year (each employee classified as stay/leave). The latter is not a forecasting problem because it deals with individual outcomes and categories, not a time-indexed projection of a single series. In some cases, classification is used for event prediction, such as predicting whether an event (equipment failure, churn, a disease outbreak) will happen in a given period.

Although this is a form of future prediction, it is not what statisticians would call 'forecasting' in the time series sense – it's a yes/no (or categorical) prediction typically handled by logistic regression or other classification models. One could transform an event prediction into a forecasting task by forecasting event counts or probabilities over time, but fundamentally if the question is 'Will event X happen (by time T)?', it is treated as a classification or survival analysis problem. Key differences between forecasting and classification:

1. **Output:** Forecasting outputs a numeric (often continuous) prediction for each future time point (or period). Classification outputs a class label (or class probability) for each instance, which may or may not have a time dimension.
2. **Inputs:** Forecasting usually uses the time index and past values (and perhaps external regressors) as input. Classification uses a feature vector for each instance (which could include time as one feature, but not necessarily a history of the label).
3. **Temporal dependence:** Forecasting explicitly accounts for temporal order and dependency; classification typically assumes independent instances (or at most uses time in a limited way, like a sequence of events in a time series classification task, which is different from forecasting future values).
4. **Use case:** Forecasting answers How much / what value will we have when...? while the classification answers "which category does this instance belong to?" or "will this event occur or not?".

To illustrate, consider anomaly detection on a time series versus forecasting (which we discuss in more detail below). If one wanted to identify whether the next data point will be an anomaly, one could frame it as a classification problem ("anomalous" vs. "normal" point) using features of recent data, which is different from forecasting the value of the next point. The classification formulation yields a probability of anomaly, whereas forecasting yields an expected value (with confidence intervals) of the next point. They address different questions.

Event prediction often overlaps with classification. For example, predicting whether a user will click on an ad in the next hour is basically a binary classification (click or not), although it has a time element ('in the next hour'). Forecasting might instead predict how many clicks in total to expect in the next hour (numeric count). So, if you are predicting whether something happens or which category it falls into, you are in classification territory, not classical forecasting.

There are also hybrid scenarios, such as forecasting the probability of an event over time. For example, one could forecast the probability that a machine fails each month for the next 12 months. That is a form of prediction (the result is a numerical probability at each future time), but it is closely related to an event prediction model. Often such problems are tackled

with time-to-event models or hazard functions rather than traditional time-series forecasts, underscoring that they straddle the line between forecasting and classification.

In summary, a classification task is not a forecasting task if the end goal is a label/category rather than a time-indexed value. If your data and question are better expressed as 'Which of these categories / outcomes will occur' than 'What will the value be on date X', you probably do not have a forecasting problem.

2.1.3 Forecasting vs. Anomaly Detection

Anomaly detection (also called outlier detection) is the identification of unusual observations or patterns that do not conform to the expected behavior of the data. In time series contexts, anomaly detection might be used to flag sudden spikes, drops, or deviations from the normal pattern. Although anomaly detection often uses time series data, it is fundamentally different from forecasting: the goal is not to predict future values but to evaluate whether current or past values are abnormal.

One way to see the difference is in the output: forecasting produces a predicted future sequence; anomaly detection produces alerts or scores for data points indicating how anomalous they are. In fact, anomaly detection can be one tool to decide when forecasting is unreliable – if an anomaly is detected, one might exclude it from model fitting or treat it specially.

Methods for anomaly detection in time series often involve comparing the observed data to an expected pattern (which could be derived from a forecast or a smoothing of past data). For example, a simple approach might forecast the next value and then if the actual comes in very different, flag an anomaly. However, more advanced anomaly detection methods do not necessarily require producing explicit forecasts; instead, they might use statistical properties or machine learning on features of the series. Hyndman et al. [55] provide an example of large-scale anomaly detection: they compute feature vectors for a large number of time series (features capturing seasonality strength, entropy, etc.) and then identify outliers in that feature space to find unusual time series. In that approach, no explicit forecasting of each series is done; they directly look for 'unusual' characteristics in historical data.

To contrast tasks, consider a server monitoring context. If you have metrics collected over time (CPU usage each minute on many servers), a forecasting problem would be: "Predict the CPU usage for the next hour for each server." An anomaly detection problem would be: 'Flag any servers that are acting abnormally right now (or in the past hour).' The latter might use the recent time series data to detect anomalies, but it is not trying to extrapolate that series into the future; rather, it is identifying deviations from expected behavior. In Hyndman et al.'s anomaly detection approach, 'we wish to identify servers that are behaving unusually' by computing features of each time series and applying outlier detection in feature space [55]. This is inherently about understanding the present/past behavior relative to normal, not about predicting the future.

Why forecasting might not be appropriate when anomaly detection is the goal: If your primary interest is to spot anomalies, applying a forecasting model might be unnecessary or even counterproductive. You could use forecasting as a component (i.e. forecast and see if actual deviates beyond a threshold), but the focus is on the deviation, not the forecast itself. Also, anomaly detection often involves unsupervised or semi-supervised techniques that do not require a forecast horizon. They look at statistical properties (mean, variance,

expected range) of recent data and flag points outside the expected range. In contrast, forecasting is a supervised learning problem (we train on past data to predict future data). Another scenario is where data are dominated by anomalies, for example, network traffic data during a cyberattack. In such a case, trying to forecast the traffic might be futile because the normal pattern is completely disrupted; instead, one should detect and respond to the anomaly. Forecasting assumes normal conditions (or conditions that can be modeled); anomaly detection explicitly targets the abnormal conditions. They serve different objectives: forecasting for planning ahead under normal variability, anomaly detection for monitoring, and alerting when unusual events occur. In the literature, the separation of these tasks is well recognized. Time-series analysis encompasses not only forecasting, but also other tasks such as anomaly detection, clustering, and classification. When a time series contains a lot of irregularities, one might first apply anomaly detection to cleanse the data or understand them, rather than directly forecasting. If anomalies are frequent, one might argue that the underlying process is not predictable enough to forecast, shifting the focus to exploratory or diagnostic analysis instead of predictive analysis.

In summary, an anomaly detection problem is not a forecasting problem because the output and intent differ: anomaly detection finds 'when / what' is strange in the data we have', while forecasting asks 'what is going to happen next'. If you find yourself more interested in identifying outlier points or sudden changes (perhaps for root cause analysis or alerting) than in predicting the next few points, then you are dealing with an anomaly detection task (or a monitoring task) rather than a forecasting task. That said, the two can interact: robust forecasting methods need to handle anomalies (often by pre-processing or modeling them separately), and conversely, forecasting methods can be used to aid anomaly detection (by providing an expected value to compare against). For example, one could forecast electricity consumption for the next hour and flag it as an anomaly if the actual consumption deviates beyond, say, 3 standard deviations of the forecast error. This is a common technique in engineering: forecast-based anomaly detection. But again, the end goal distinguishes the tasks: if the end goal is the flag / alert, it is anomaly detection; if the end goal is the predicted value, it is forecasting.

2.1.4 Forecasting vs. Regression (General Predictive Modeling)

At first glance, forecasting might seem like just a special case of a regression problem – after all, we are predicting a numerical value (continuous output), which is what regression does. In fact, in machine learning terms, time series forecasting is indeed often treated as a regression problem where the features are time-indexed (lags of the series, time indicators, etc.) and the target is the future value.

The crucial difference is the temporal order and dependence of the observations. In a standard regression problem, we assume that training examples are independent (IID assumption). For example, you might have a dataset of houses with features (size, location, etc.) and target (price), and you fit a regression model to predict price from features. You can randomly shuffle the data during training; there is no inherent order.

Time-series forecasting violates this assumption because each 'example' (time step) is related to previous ones. You cannot shuffle time series data without destroying the very structure you need to model. So regression models can be used for forecasting, but they must be applied in a time-aware way (often called time series regression). A simple connection between the two is: 'A regression problem in which the input variables are ordered by time is called a time series forecasting problem.' In other words, if you take a

regression model and feed it lagged values of a time series as input to predict the next value, you are essentially doing time-series forecasting via regression. The mathematics might be similar (minimizing errors), but the problem framework and validation differ. Forecasting requires respecting the time order (no peeking into future values during model training) and often updating the model as new data arrive. One way to distinguish forecasting from generic regression is by asking the question being asked. In forecasting, the question is: What will y be at time $t+h$ given what we know up to time t? In regression, a typical question is 'What is y for a given set of characteristics X?' – There is no requirement that these features come from past values of y or that there's a time gap. In forecasting, we usually use autoregression (past values of y as characteristics) or exogenous variables that themselves can be predicted or assumed known. In regression, we often have a fixed data set and are concerned with associations at one point in time or in a cross-sectional sense.

When is a regression not a forecasting problem? If you have a regression setup where the target is continuous but not a future value of a time series, then it is not forecasting. For example, predicting a person's weight from their height and age is a regression problem, not a prediction, because you are not extrapolating their weight over time; you are just mapping characteristics to an outcome. Even if you collect such data over time from many people, the regression itself does not use the temporal ordering of weight for one individual; it uses different individuals or conditions as data points.

On the other hand, if you treat "time" as just another input in a regression model (such as the year as a feature to predict some value), you are implicitly making a forecast if you extrapolate beyond the range of the data. However, regression models without explicit time series structure might fail to capture things like autocorrelation. Traditional forecasting methods (ARIMA, exponential smoothing) incorporate the idea that residuals should be uncorrelated and use lagged terms to account for correlation. A naive regression that ignores these could yield overly optimistic results (because it might effectively use future information by not accounting for serial correlation in residuals).

Another aspect is evaluation and training: In forecasting, one must take care to train on past data and evaluate on future data (e.g., using a rolling origin evaluation or holdout of the last h points), whereas in general regression tasks, one might do random train-test splits. For forecasting, random splits are usually invalid because they mix past and future. So, the modeling and validation process is specialized. In summary, forecasting is a subset of regression problems, one that involves time-ordered data and often autoregressive modeling. All forecasting problems can be seen as regression problems (predicting a number), but not all regression problems are forecasting problems. The telltale sign is the dependency on the previous output and the focus on future (out-of-sample) time points. If your regression problem can be reframed as 'predict y_{t+1} from $y_t, y_{t-1}, ...$ etc.' or similar, then it is a forecasting problem. If it cannot (because the target is not explicitly a future time-indexed value or because the inputs are not past values of the same series), then you are dealing with a general regression problem, not a time-series forecast. In practice, forecasters often use regression models within forecasting (for example, adding regression terms for holidays or using machine learning regression algorithms on lagged features), but they always respect the temporal ordering of data and evaluate on future time points. As a result, forecasting can be seen as a specialized regression problem with a sequential dependency constraint. When that constraint is not present, one should not force a forecast interpretation.

2.1.5 Definition and Criteria

Forecasting is the process of predicting future values of a time series based on historical data. It assumes that historical patterns, such as trends and seasonality, will persist.

Use forecasting when:

- Data are sequential and time-indexed
- Temporal patterns (e.g., trends, cycles, autocorrelation) are present
- Stationarity or transformations to stationarity are feasible

Avoid forecasting when:

- The target is a class label (classification)
- No meaningful patterns or too many structural breaks exist
- You're detecting rare or unusual events (use anomaly detection instead)

2.1.6 Forecasting vs. Other Methods

Task	Goal	Time Series?	Output Type
Forecasting	Predict future values	Yes	Numeric (continuous)
Regression	Predict from features	Optional	Numeric
Classification	Assign labels	Often no	Categorical
Anomaly Detection	Flag irregularities	Often yes	Boolean/Score

2.1.7 Where Forecasting Excels

Examples of domains where forecasting methods are particularly effective.

1. **Demand Planning:** Forecasting is indispensable in demand planning for businesses. When customer demand follows relatively stable patterns (trend growth or seasonal cycles), forecasting can predict future sales with reasonable confidence. This empowers companies to manage production, staffing and supply procurement proactively. For example, retail sales often exhibit seasonality (holiday peaks, summer slowdowns) that forecasting models capture to ensure shelves are stocked appropriately. In such settings, forecasts directly inform operational decisions, and even moderate accuracy improvements can translate into substantial cost savings and service-level improvements.
2. **Supply Chain:** In supply chain management, accurate demand forecasts enable optimal inventory and capacity planning, which in turn reduces costs and improves service. Companies with more accurate forecasting methods have been shown to reduce inventory holding costs by up to 20–50%, since they can avoid overstocking and understocking. A well-known example is forecasting for intermittent, slow-moving items (such as spare parts or low-volume products). Traditional methods struggle with lumpy demand, but specialized forecasting techniques help here. Croston's method (Croston, 1972) [22] is a classic forecasting approach tailored to intermittent demand. Updates estimates of the size and interval of the demand only when a nonzero demand occurs, effectively smoothing sporadic demand over time. Studies in operations research have found that the Croston method (and its variants) can outperform simple exponential smoothing in intermittent demand series, leading to more accurate forecasts. This improved forecast accuracy translates into tangible benefits: manufacturers and distributors can hold the right amount of safety stock and reorder at the right times, minimizing stock shortages while avoiding excess

inventory. In practice, implementing such forecast-driven inventory optimization has yielded significant cost reductions along with higher fill rates (product availability). Thus, supply chains greatly benefit from forecasting, especially in balancing supply with uncertain demand, a core challenge where better predictions directly equate to efficiency and profit.

3. **Energy Load Forecasting:** Electricity demand exhibits a strong temporal structure: daily and weekly cycles, seasonal effects, and weather dependencies. Forecasting is critical for grid management, pricing, and operational planning. Utilities rely on models that incorporate temperature, time-of-day, day-of-week, and calendar features to predict future load with high accuracy. Accurate load forecasts help balance supply and demand, reduce the dependence on costly reserve capacity, and support the integration of renewable energy sources.

- **Demand Planning:** Sales, inventory management
- **Energy Load Forecasting:** Electricity usage with seasonal effects
- **Finance:** Predicting volatility and returns

To summarize this section: Forecasting is distinct in that it predicts a future numerical value (or distribution of values) for a time-dependent process, whereas classification predicts a class/category, anomaly detection finds irregularities, and general regression predicts a value but not necessarily in a time-linked way. Each approach has its place. A problem qualifies as a forecasting problem when time dynamics is central and we seek to extrapolate the time series; it does not qualify when the primary output is a category or when time order is irrelevant or disrupted.

2.1.8 The Business Impact of Forecast Accuracy

Although identifying forecastable time series is crucial, an equally important practical consideration is the value of improving forecast accuracy. Beyond theoretical metrics such as RMSE or MAPE, forecast accuracy directly affects business performance and financial outcomes.

2.1.8.1 Sales and Revenue Growth

Higher forecast accuracy leads to fewer stockouts and missed opportunities, capturing demand that could otherwise be lost. Harvard Business Review reports that stockouts cause an average loss 4% in sales, which means that for a company with $1 billion in annual revenue, approximately 40 million could be at stake [82]. McKinsey research finds that AI-driven forecast improvements can reduce product unavailability by up to 65%, leading to higher sales capture and customer loyalty [20].

2.1.8.2 Profit Margins and Inventory Cost Reduction

Improved forecasting also minimizes overstock, avoiding costly markdowns and spoilage. Inventory carrying costs typically range between 20–30% of inventory value per year [39]. Reducing excess inventory, thus, significantly boosts margins. For example, a packaging manufacturer improved forecast accuracy by 20 percentage points and eliminated excess inventory of 1 million, while simultaneously improving its order-fill rate from 97.7% to 98% [40].

2.1.8.3 Working Capital and Financing Cost Savings

Inventory reduction through better forecasting frees up cash, directly reducing financing costs or opportunity costs of capital tied to stock. If a company reduces its investment in

inventory by $10 million and its capital cost is 10%, this translates to immediate savings of $1 million. Improved forecasting also shortens the cash conversion cycle, improving liquidity and financial resilience [19].

2.1.8.4 Stockout Cost Avoidance and Customer Satisfaction

Stockouts not only cause immediate lost sales, but can trigger long-term customer churn and brand reputation damage. Better forecasts prevent costly emergency actions such as delayed shipment or late-night production overtime [82]. Maintaining high service levels through better forecasting has been shown to improve customer retention, repeat sales, and even allow companies to maintain pricing power [20].

2.1.8.5 Quantified Real-World Benefits

Company / Study	**Forecast Improvement**	**Benefit**
Global Beverage Producer	Advanced demand forecasting	Saved **$9M annually** via production optimization [56]
McKinsey (Retail / CPG)	10–20% forecast accuracy increase	**5% inventory cost reduction**, **2–3% revenue uplift** [20]
IBF Survey (Tech Firms)	1% improvement in forecast error	Saved **approximately $1M–$1.5M annually** [14]
Kraft Heinz	8% forecast accuracy increase	Improved supply chain efficiency and service [71]

Table 2.1: Examples of forecast accuracy improvements and their financial benefits across industries.

These examples highlight that forecast accuracy is far more than an academic concern; it has direct, measurable, and often multi-million-dollar consequences for organizational performance. Enhancing forecast accuracy not only improves predictive metrics, but also delivers substantial business value by driving revenue growth, reducing costs, optimizing working capital, increasing customer satisfaction, and strengthening competitive advantage.

2.2 Why Forecastability Matters

So how can we tell if a series is predictable before investing effort in modeling? IN Chapter 1, we explore how forecasting evolved from mystical divination to scientific modeling. But before we dive into modeling techniques, we need to face a more fundamental question: how forecastable are the data in front of us? Forecastability is not about what model you use —it is about whether there is anything to model in the first place.

Forecastability is a property of the time series itself. It captures the theoretical limit of how well future values can be predicted given the available information. If a time series is completely random, no amount of clever modeling will improve your forecast. If it holds structure, signal, or memory, then the door to predictability opens.

Understanding forecastability helps you answer critical questions.

1. Is this series worth modeling?
2. Should we spend time feature engineering or just use a naive baseline?
3. What is the best achievable accuracy we can hope for?
4. Can we set expectations with stakeholders before burning cycles in modeling?

forecastability comes from forecasting volatility instead of returns: volatility (variance of returns) often has long memory and predictable structure (e.g. GARCH effects), so one might say the volatility process has higher intrinsic forecastability even if price levels do not. Overall, in economics and finance, forecastability is assessed by the degree to which past data or other predictors can explain variance in the future – usually this is very low for asset prices (often no better than coin toss beyond short horizons), whereas macroeconomic series (like GDP growth or inflation) may have moderate forecastability due to business cycles, autocorrelation, and causal drivers.

In summary, while terminology may differ, e.g., 'predictability', 'forecastability', 'signal vs. noise', all disciplines seek to measure how much of the variation of a time series is explainable and not purely random.

Next, we review the key metrics and theoretical measures that have been developed to quantify forecastability.

This section surveys key metrics across different theoretical foundations—statistical, spectral, information-theoretic, and algorithmic. Each measure is explained in terms of its definition, interpretation, and relevance to forecastability. The goal is to provide consistent, comparable insights into how each diagnostic assesses the potential predictability of a time series.

2.4 Key Measures of Forecastability

How can we assess whether a time series is predictable before investing substantial effort in modeling? Before committing to a full forecasting pipeline, analysts can apply quantitative diagnostics to evaluate the forecastability of a series. This section introduces a range of measures—from classical statistics to modern machine learning tools—that objectively quantify a series' predictability.

By using these metrics, we can assess whether a time series carries a meaningful structure (recall: a pure noise series has no useful signal for forecasting) and decide whether sophisticated modeling is justified. We describe both traditional techniques (e.g., entropy-based metrics, autocorrelation tests) and practical forecasting skill scores, while also highlighting newer toolkits that automate forecastability analysis. This blended approach bridges theory, the limits of predictability, with practice: hands-on tests to check those limits.

Forecastability refers to the inherent predictability of a time series, essentially how well future values can be inferred from the past. Unlike forecasting methods, which focus on how to generate predictions, forecastability is a property of the data itself. Quantifies the theoretical limit of the prediction accuracy achievable, independent of any specific method [9].

Highly forecastable series exhibit strong patterns or structures that can be learned, while poorly forecastable series are dominated by randomness or noise. Evaluating forecastability is critical: It helps determine whether sophisticated models are worthwhile or whether a series is so noisy that even the best models will yield large errors.

In what follows, we review metrics used to evaluate time series forecastability. We cover classical approaches (e.g., autocorrelation and variance-based measures from statistics and econometrics) as well as modern information-theoretic and complexity-based measures (e.g., entropy, algorithmic complexity) from machine learning and nonlinear dynamics. Importantly, we focus on the evaluation of the predictability itself, the predictability of the

series, not on specific forecasting algorithms.

Researchers have devised a wide range of metrics to quantify the predictability of a time series. These measures span from simple statistical descriptors (such as autocorrelation or coefficient of variation) to sophisticated information-theoretic quantities (such as entropy rates and mutual information). Broadly, we can categorize forecastability measures into model-based (realized) metrics, which depend on a particular forecasting approach, and intrinsic metrics, which attempt to characterize predictability independent of any specific model. In the following, we survey key measures, explaining which aspect of "predictability" each captures. We also note how different measures are applied in practice in domains such as demand, energy, and finance.

2.4.1 Coefficient of Variation (CV) and ADI-CV Framework

In demand forecasting, particularly within supply chain contexts, the ADI (Average Demand Interval) and CV (Coefficient of Variation) framework proposed by Syntetos and Boylan [90] provides a practical method for classifying time series based on their forecastability. This classification enables practitioners to select appropriate forecasting models in advance.

The formulas are:

$$\text{ADI} = \frac{n}{n_z}$$
$$\text{CV} = \frac{\sigma}{\mu}$$

where:

- n = total number of periods
- n_z = number of periods with nonzero demand
- μ = mean demand
- σ = standard deviation of demand

The four demand types are:

- **Smooth:** Low ADI and low CV. Highly predictable demand patterns.
- **Intermittent:** High ADI, low CV. Forecastable with specialized methods such as Croston's, ADIDA, or IMAPA.
- **Erratic:** Low ADI, high CV. High uncertainty, often driven by price sensitivity or promotions.
- **Lumpy:** High ADI and high CV. Extremely difficult to forecast, often related to slow-moving or event-driven items.

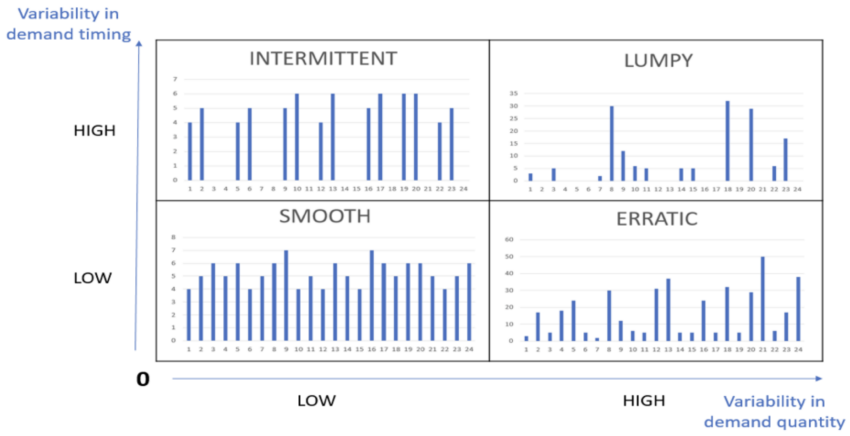

Figure 2.1: Demand Type Classification Based on ADI and CV.

2.4.2 Variance Ratio and Theoretical Predictability Limits

One fundamental notion of forecastability is the fraction of variance that is predictable. If we denote the variance of the time series by σ_y^2 and the variance of the unavoidable forecast error (i.e. the error of an optimal forecasting method or oracle) by σ_e^2, then a basic predictability index can be defined as follows:

$$\text{Forecastability} = 1 - \frac{\sigma_e^2}{\sigma_y^2}$$

In other words, this is the proportion of the variance of the series that can be explained by the best possible forecast [9].

The concept of using explained variance to measure forecastability was explicitly proposed by Granger and Newbold [38]. It defines a **theoretical upper bound on the R^2 of any forecasting model**.

Here, σ_y^2 is the variance of the observed time series and σ_e^2 is the variance of the forecast error from an optimal model.

- If the series is *pure noise*, then $\sigma_e^2 \approx \sigma_y^2$; the forecast explains no additional variance, so the forecastability approaches **0**.
- If the series has a *strong deterministic pattern* (e.g. trend or seasonality) with minimal noise, then $\sigma_e^2 \ll \sigma_y^2$, and forecastability approaches **1**.

Although the true value of σ_e^2 is typically unknown (since we rarely have access to an oracle model), this framework informs many practical metrics.

One such example is the **Nash–Sutcliffe Efficiency (NSE)**, also known as the *Coefficient of Efficiency (CE)*, widely used in hydrology. NSE is defined as:

$$\text{NSE} = 1 - \frac{\text{MSE}}{\sigma_y^2}$$

It evaluates how well a forecasting model performs relative to a naive benchmark (e.g., using the mean of the series as a constant forecast):

- **NSE = 1** indicates a perfect forecast (all variance explained),
- **NSE = 0** means the forecast is no better than the mean,
- **NSE < 0** suggests the model performs worse than the naive mean forecast.

Although NSE is a performance measure realized for a given model, it reflects the same underlying idea of variance-based forecastability. In practice, researchers approximate intrinsic forecastability by fitting highly flexible models. For example, Wang et al. [97] estimated the predictability of daily streamflow by applying ARMA models and computing the variance ratio as a proxy for the optimal forecast error.

Another useful approach to measuring forecastability involves comparing a model's performance on the original time series versus a randomly shuffled version. Kaboudan [57] introduced a predictability score based on this idea using genetic programming. The logic is simple: If a time series has meaningful temporal structure, a forecasting model should perform better on the original data than on a version with the same values but random order. A score significantly less than 1 indicates the presence of an exploitable structure; a score close to 1 suggests little or no forecastable pattern, similar to i.i.d. noise [9].

This idea is closely related to skill scores commonly used in meteorology, where forecasts are evaluated relative to a naive benchmark (e.g., assuming tomorrow equals today). If a complex model performs only marginally better than such a baseline, the series is likely to have low forecastability.

It is important to note that variance-based measures, such as R^2 or Nash-Sutcliffe efficiency, typically assume optimality under a least-squares criterion and are best suited to capture linear patterns. However, many real-world time series contain non-linear dynamics or regime changes that these measures may miss. That's why information-theoretic approaches (discussed later) are often used to detect more complex dependencies.

However, variance-based diagnostics offer a useful starting point. In demand planning, for instance, the coefficient of variation (CV) is a practical proxy for the signal-to-noise ratio: a low CV implies more predictable demand, while a high CV suggests greater randomness. These simple metrics underpin many of the rule-of-thumb forecastability assessments used in practice.

2.4.3 Autocorrelation Function (ACF): Measuring Linear Predictability

Autocorrelation is a fundamental concept in time series analysis. It measures the correlation between a value in the series and a lagged version of itself, that is, how well the series "remembers" its previous values. Formally, the autocorrelation at lag k, denoted $\rho(k)$, is the Pearson correlation between y_t and y_{t-k}.

- If $\rho(1) \approx 0.9$, the series exhibits strong persistence: Each observation is highly correlated with the previous one, making it easier to forecast using simple methods such as a persistence model (i.e., predicting the next value will be equal to the current one).
- If $\rho(1) \approx 0$, the series has no memory (as in white noise), and past values do not provide useful information for prediction.

A basic measure of linear forecastability is the magnitude of the first autocorrelation:

$$|\rho(1)|$$

A value close to 1 implies high forecastability, while a value close to 0 suggests poor forecastability using linear models.

To assess the structure beyond the first delay, analysts commonly examine the **autocorrelation function (ACF)**, which plots $\rho(k)$ over multiple delays. A slowly decaying ACF indicates long memory or seasonality and generally suggests a predictable series, though it may also imply non-stationarity. In contrast, a flat ACF, where autocorrelations are near zero at all lags, indicates that the series behaves like white noise.

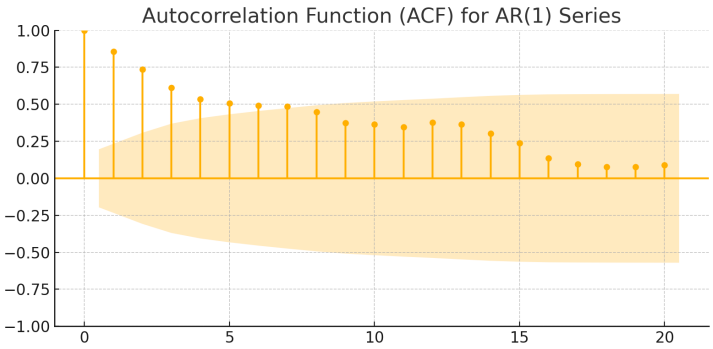

Figure 2.2: Example of an autocorrelation function (ACF) plot. Strong autocorrelations at low lags indicate persistence and high forecastability.

As Box and Jenkins [12] emphasized, if autocorrelations are close to zero at all lags, the series can be treated as white noise, which means that no structure can be exploited for forecasting and the best predictor is simply the historical mean.

Summary:

- High autocorrelation with low delays ⇒ strong memory ⇒ good short-term forecastability.
- Slow ACF decay ⇒ long memory or seasonality ⇒ potentially predictable, but may require transformation.
- Flat ACF ⇒ no structure ⇒ poor forecastability.

2.4.4 Partial Autocorrelation and Model Parsimony

While the ACF reveals correlation at increasing lags, the Partial Autocorrelation Function (PACF) isolates the unique contribution of each lag, removing the influence of intermediate terms. Closely related to autocorrelation is the concept of partial autocorrelation, which helps to identify the most relevant lag terms for linear models like ARIMA. If a time series can be accurately captured by a low-order ARIMA model, that is, with just a few lag terms and low residual variance, it suggests high forecastability. In contrast, if even the best-fitting AR(p) model leaves residuals with variance nearly as large as the original series, the data likely contain little linear structure to exploit.

Hyndman et al. [59] have highlighted the importance of simple characteristics, such as autocorrelation at lag 1 (ACF1), in assessing the predictability of time series. These features have been used in time series classification frameworks [79] to distinguish between easy and difficult series. For example, analysis of M4 forecast competition showed that many of the hardest-to-predict series had low autocorrelation and highly erratic or intermittent patterns, often resembling high-frequency noise.

2.4.5 Variance Ratio as a Measure of Forecastability

Another useful linear diagnostic for forecastability is the **variance ratio test**, which examines how the variability of a time series evolves across different time horizons. The idea is grounded in the behavior of a *random walk*, where the variance of k-step returns should grow linearly with the number of steps. That is, for a true random walk:

In contrast, if the k-step variance is *less than* k times the one-step variance, the series exhibits **mean-reverting behavior**—values tend to return to a long-term average, which can be exploited for forecasting. If the k-step variance is *greater than* expected, the series may be **trending**, implying directional drift and potential long-term predictability.

This relationship is captured by the **variance ratio (VR)**:

$$\mathrm{VR}(k) = \frac{\mathrm{Var}(y_{t+k} - y_t)}{k \cdot \mathrm{Var}(y_{t+1} - y_t)}$$

Interpretation:

- $\mathrm{VR}(k) \approx 1$: The series behaves like a random walk — no forecastable structure.
- $\mathrm{VR}(k) < 1$: The series is mean-reverting — deviations tend to self-correct.
- $\mathrm{VR}(k) > 1$: The series is trending — persistence implies potential forecastability.

This approach is widely used in econometrics, particularly in analyzing financial time series. The *Lo and MacKinlay variance ratio test* is a formal implementation of this idea and is often applied to assess whether asset prices follow a random walk, as posited by the Efficient Market Hypothesis.

Illustrative Examples:

- *Mean-reverting:* Stock market volatility often spikes and then reverts to the normal variance ratio < 1.
- *Trending:* Macroeconomic variables such as inflation or GDP growth can exhibit persistent trends—variance ratio > 1.
- *Random walk:* High-frequency exchange rate returns typically yield $\mathrm{VR}(k) \approx 1$.

Detecting significant deviations of $\mathrm{VR}(k)$ from 1 provides evidence of a forecastable structure, especially over multistep horizons where short-term autocorrelations may not reveal deeper patterns.

In summary, autocorrelation-based metrics capture linear predictability – the degree to which future values can be predicted by linear projections of the past. They are easy to compute and interpret. However, purely nonlinear or context-dependent patterns will not be reflected in simple ACF statistics. Thus, while a strong autocorrelation is a sufficient indicator of some forecastability (especially short-term), a weak autocorrelation does not guarantee unpredictability – the series might have nonlinear dependencies that require other measures to detect.

2.4.6 Information-Theoretic Measures (Entropy and Predictability)

Information theory, introduced by Claude Shannon, offers a powerful framework for quantifying uncertainty and information in dynamical systems. At its core is **Shannon entropy**, which measures the average uncertainty (or "surprise") associated with the outcomes of a random variable.

For a discrete variable X with possible outcomes $\{x_1, x_2, \ldots, x_n\}$ and probability distribution $P(X)$, the Shannon entropy is defined as:

$$H(X) = -\sum_i P(x_i) \log_2 P(x_i)$$

This value reflects the disorder or unpredictability of the system. For example:

- A fair coin has $P(\text{heads}) = 0.5$, yielding $H = 1$ bit — maximal uncertainty.
- A biased coin with $P(\text{heads}) = 0.9$ has lower entropy, around $H \approx 0.47$ bits, reflecting greater predictability.

2.4.6.1 Entropy Rate for Time Series

When analyzing temporal data such as financial markets, physiological signals or climate variables, we generalize the entropy to the concept of the **entropy rate**, which measures the average uncertainty per time step in a stochastic process.

For a time series $\{X_t\}$, the entropy rate is defined as:

$$H_{\text{rate}} = \lim_{T \to \infty} \frac{1}{T} H(X_1, X_2, \ldots, X_T)$$

This accounts for dependencies between observations. Some examples illustrate the concept.

- *Maximum entropy rate*: A memoryless process (e.g., independent coin flips) has $H_{\text{rate}} = H(X)$, since each new value is fully independent.
- *Low entropy rate*: A perfectly regular or deterministic process, such as $x_t = \sin(t)$, has $H_{\text{rate}} \approx 0$ — future values are almost completely determined by past values.

Entropy Rate as a Fundamental Limit to Forecasting

The **entropy rate** acts as a theoretical ceiling on forecasting accuracy—similar to a physical law. It quantifies the average uncertainty (in bits per step) in a time series and directly determines how predictable the process can be, regardless of the forecasting method or computational power available.

Understanding Entropy Rate Intuitively

Entropy rate measures how much new information each data point carries, given the past. The more "surprise" or randomness per step, the harder it is to predict what comes next. Two illustrative extremes help clarify:

- **Low entropy rate (e.g., 0.2 bits/step)**: A metronome ticking at regular intervals. Each tick is fully determined by the one before—there is almost no new information, making the process highly predictable.
- **High entropy rate (e.g., 3 bits/step)**: Cryptographic random numbers. Each number is independent of the past, yielding maximum surprise. Even with complete knowledge of prior values, the next is unpredictable.

The Predictability Bound

This intuition can be formalized. For a system with:

- N: number of possible states (e.g., 2 for a coin, 6 for a die),
- H_{rate}: entropy rate in bits per time step,

the **maximum achievable prediction accuracy** $\Pi_{\max}$ is bounded by:

$$\Pi_{\max} \leq 1 - \frac{H_{\text{rate}}}{\log_2 N}$$

As entropy increases toward its maximum value $\log_2 N$, predictability approaches zero. Conversely, when entropy is near zero, forecasts can be nearly perfect.

Worked Examples

System	N	H_{rate} (bits/step)	$\Pi_{\max}$
Fair coin toss	2	1.0	$1 - \frac{1.0}{\log_2 2} = 0\%$
Loaded die*	6	1.5	$1 - \frac{1.5}{\log_2 6} \approx 41\%$
Periodic signal	100	0.01	$1 - \frac{0.01}{\log_2 100} \approx 99\%$

Table 2.3: Examples of entropy rate and resulting theoretical predictability. *Loaded die: non-uniform but not perfectly deterministic.

Key Implications

- **Perfect prediction** is only possible when $H_{\text{rate}} = 0$ (pure determinism). Example: Predicting sunrise using celestial mechanics.
- **Zero predictability** occurs when $H_{\text{rate}} = \log_2 N$ (maximal randomness). Example: Guessing lottery numbers.
- **Real-world examples:**
  - *Weather forecasting:* $\Pi_{\max} \approx 60$–80% — chaotic but partially deterministic.
  - *Stock prices:* $\Pi_{\max} \approx 10$–30% — high entropy due to external and stochastic factors.

Why This Matters

This inequality helps explain why:

- Weather forecasts degrade over time — chaos accumulates entropy.
- Financial markets resist prediction — high entropy reduces signal strength.
- Data compression is possible — low entropy signals are more predictable and redundant.

In summary, the entropy rate is not just a statistic—it defines the theoretical limit of what any forecasting model can achieve.

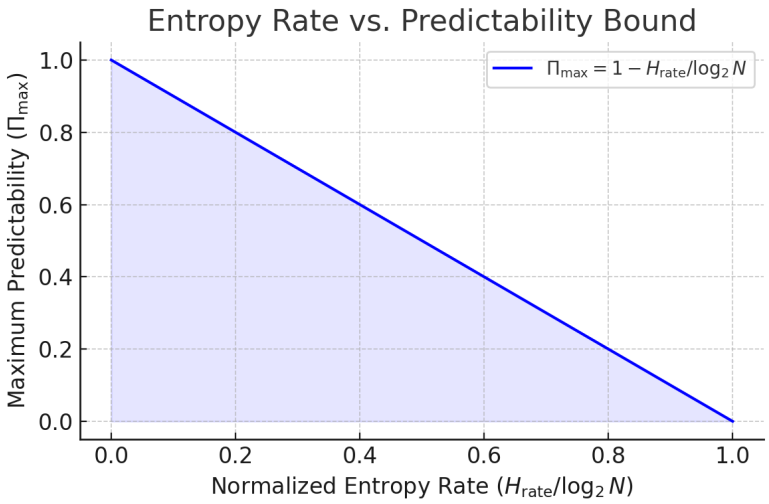

Figure 2.3: As normalized entropy rate increases, the maximum theoretical predictability decreases. At full entropy ($H_{\text{rate}} = \log_2 N$), prediction becomes no better than random guessing.

Practical Implications

Estimating the entropy rate of a time series allows us to assess its inherent forecastability.

- **Chaotic systems** (e.g., weather): Moderate entropy rates. Predictable over short horizons, but diverge over time.
- **White noise**: Maximum entropy. No structure to exploit. $\Pi_{\max} = 0$.
- **Periodic or deterministic signals**: Low entropy. Predictability is almost perfect. $\Pi_{\max} \approx 1$.

In practice, entropy rates can be estimated using methods such as Lempel-Ziv compression [103], model-based techniques, or permutation entropy. These tools are used in fields such as climatology, neuroscience, economics, and engineering to assess the limits of forecasting and guide model selection.

A seminal example of entropy-based predictability analysis is the work by Song et al. [88], which studied human mobility patterns. They modeled a person's sequence of visited locations as a time series of discrete states and calculated its Shannon entropy. Using **Fano's inequality**, they derived an upper bound on the accuracy with which any algorithm — regardless of complexity — could predict the next location of the individual. Surprisingly, despite the apparent randomness of human movement, they found that theoretical predictability was as high as $\sim 93\%$. This suggests that human mobility, while noisy, is highly structured.

One of the key challenges in applying Shannon entropy to real-world time series is estimating the joint distribution over long sequences, which is often computationally infeasible.

2.4.6.2 Lempel–Ziv Complexity (LZC)

Lempel–Ziv Complexity (LZC) is an algorithmic measure of the complexity and unpredictability of a time series. It estimates how much new information is generated as a sequence progresses by measuring the rate at which unique patterns (or substrings) appear.

Intuitively, a highly repetitive sequence (e.g., 010101...) is highly compressible and thus

has low complexity. In contrast, a random sequence (e.g., generated by coin tosses) is not compressible and has high LZC.

Computation: LZC works by scanning a sequence and counting the number of new subsequences encountered during parsing. It does not require assumptions of stationarity or linearity and can be applied to symbolic, binary, or discretized continuous series.

Forecasting implications: - A low LZC value implies the series has many repeated patterns and is therefore more predictable. - A high LZC suggests a high level of novelty or disorder, reducing the chances that past patterns can inform future values.

Applications: LZC has been used in mobility modeling, EEG signal classification, market data analysis, and entropy estimation for human behavior. It is the basis for many compression-based entropy estimates, and approximates the entropy rate for symbolic sequences.

Reference: [103], [88]

2.4.6.3 Approximate Entropy (ApEn) – Measuring Regularity

Approximate Entropy (ApEn) was introduced by Pincus [78] as a statistic to quantify regularity and unpredictability in time series data. In essence, ApEn measures the likelihood that if we see a certain pattern of observations, a slightly longer sequence will deviate from that pattern.

In simpler terms, it asks: How often do similar patterns repeat (versus surprise us by diverging)? A low ApEn value indicates that the series is very regular: If two segments of length m match closely, their next values are usually also similar. A high ApEn means that the series is irregular – even if you find two segments that look like m points, the $(m+1)$th points might be completely different.

Formally, ApEn is calculated by counting, for each subsequence of length m, the fraction of other subsequences that remain within a tolerance r; then the same for length $m+1$ and the logarithmic difference. The result ($\text{ApEn}_{m,r}$) is lower when the time series has more repeated patterns (hence more short-term predictability) and higher when every sequence of observations is followed by many different outcomes (less predictability).

Advantages ApEn is straightforward to compute and works on short and noisy data (it was originally applied to medical signals like heart rates). It is a fast, single-number summary of how predictable the process is on a small scale. It can capture non-linear regularity that traditional metrics (like variance or autocorrelation) might miss. For example, a random sequence has a high ApEn (patterns do not repeat), whereas a sinusoidal sequence has ApEn near 0 (highly regular).

Limitations: ApEn has a known bias: it counts self-matches (each subsequence trivially matches itself), which causes ApEn to underestimate entropy especially for short series. It also requires choosing the parameters m (pattern length) and r (tolerance), and the results can be sensitive to these choices and to the length of the data. In practice, one must ensure the time series is long enough (relative to m) and pick r (often as a percentage of the standard deviation of the data) appropriately.

Despite these limitations, ApEn provides a useful a priori indicator of forecastability: if ApEn is extremely high (close to the maximum for given m), the series may be essentially as good as random in terms of short-term patterns, warning us that achieving accurate

forecasts will be difficult. If ApEn is low, the series has a lot of structure that a model might exploit. However, ApEn is not the final word – it led to improved measures such as sample entropy.

Python Example – Computing ApEn: The following code computes ApEn for a time series (using $m = 2$, $r = 0.2\sigma$) to illustrate how it distinguishes a periodic series from a random series:

```
import numpy as np
import plotly.graph_objects as go

# Approximate Entropy function
def approximate_entropy(series, m, r):
    N = len(series)
    tol = r * np.std(series)

    def _phi(m):
        C = []
        for i in range(N - m + 1):
            seq_i = series[i:i + m]
            count = 0
            for j in range(N - m + 1):
                if np.max(np.abs(series[j:j + m] - seq_i)) <= tol:
                    count += 1
            C.append(count / (N - m + 1))
        return np.mean(np.log(C))

    return _phi(m) - _phi(m + 1)

# Generate data
np.random.seed(0)
data_periodic = np.sin(np.linspace(0, 2 * np.pi * 5, 100))
data_random = np.random.randn(100)

# Compute entropies
entropy_periodic = approximate_entropy(data_periodic, m=2, r=0.2)
entropy_random = approximate_entropy(data_random, m=2, r=0.2)

# Plot using Plotly
fig = go.Figure()
fig.add_trace(go.Scatter(
    y=data_periodic,
    mode='lines',
    name='Periodic Signal (ApEn ~ {:.2f})'.format(entropy_periodic),
    line=dict(color='blue')
))
fig.add_trace(go.Scatter(
    y=data_random,
    mode='lines',
    name='Random Signal (ApEn ~ {:.2f})'.format(entropy_random),
    line=dict(color='red')
))
fig.update_layout(
    title='Periodic vs. Random Time Series with Approximate Entropy',
    xaxis_title='Time Index',
    yaxis_title='Value',
    legend=dict(x=0.01, y=0.99),
    template='simple_white'
)
fig.show()
```

Listing 2.1: Python code for approximate entropy and visualization

As shown in Figure 2.4, the periodic series has significantly lower ApEn compared to the random series, indicating higher forecastability.

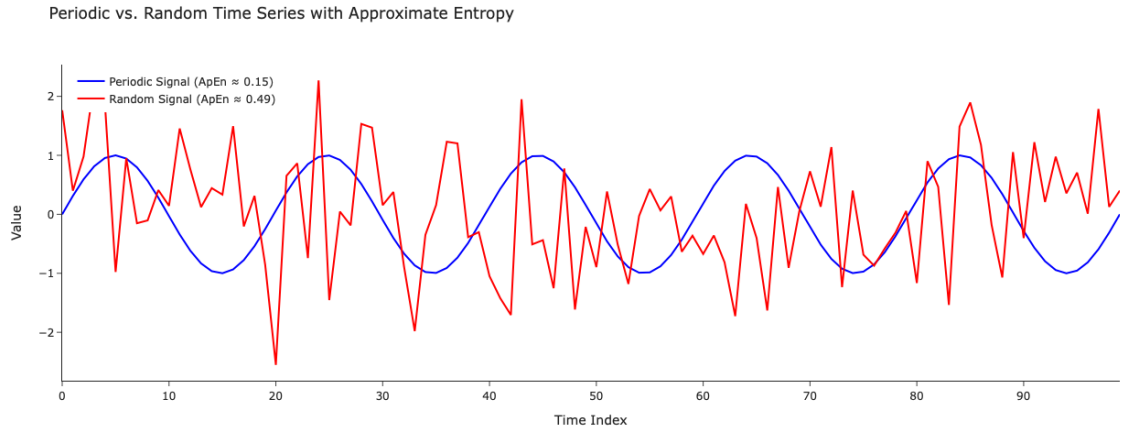

Figure 2.4: Approximate entropy of a periodic signal (low entropy) versus random signal (high entropy). Low ApEn indicates regularity and high short-term forecastability.

Another widely used information measure is **Permutation Entropy (PE)**. Introduced by Bandt and Pompe (2002), permutation entropy looks at the order relations of consecutive values in the time series [5]. For example, for an embedding dimension d (length of pattern), one looks at sequences of d successive points and records the ordinal rank pattern (e.g. x_{t+d-1} is the largest, x_t is the second smallest, etc.). There are $d!$ possible ordinal patterns. The permutation entropy is the Shannon entropy of the distribution of these ordinal patterns [9]. If a time series is completely random, all order patterns are (approximately) equally likely, leading to a high permutation entropy (close to $\log(d!)$). If the series has structure (say always increasing trends of a certain length), some patterns will be much more common, and others ("forbidden patterns") might never occur, yielding a lower entropy. Permutation entropy has become popular because it is simple, fast to compute, and invariant under monotonic transformations (use only rank order). It effectively measures the complexity of the dynamics of the time series. A low permutation entropy indicates a more predictable series. For example, in financial market analysis, researchers found that major stock indices exhibit certain forbidden ordinal patterns far more often than pure chance would allow. This indicates underlying structure in price movements (albeit subtle), and the normalized permutation entropy was proposed as a model-independent inefficiency (predictability) measure. In fact, the presence of forbidden patterns in asset prices suggests a deviation from randomness that could be used for forecasting.

2.4.7 Sample Entropy (SampEn)

Sample Entropy (SampEn) is a refinement of ApEn proposed by Richman and Moorman [83] to address ApEn biases. SampEn uses a similar conceptual approach to ApEn – quantifying the probability that two sequences of length m that match (within tolerance r) will also match at the next point — but makes two critical changes in the algorithm:

1. it excludes self-matches

3. Forecasting Metrics

> *"Don’t sharpen your ruler before checking if there’s anything to measure."*
>
> — ***Forecasting Proverb***

Chapter 2 emphasized that before forecasting begins, we must ask whether a time series is forecastable at all — whether it contains signal strong enough to rise above noise. It introduced the concept of forecastability not as a modeling choice, but as a property of the data itself, grounded in entropy, variability, autocorrelation, and temporal structure.

The chapter 2 details diagnostic tools such as CV-ADI classification, permutation entropy, and spectral analysis, showing how they can be used to segment series, model classification strategies, and set realistic expectations. It also underscored a crucial truth: no metric, however precise, and no model, however powerful, can retrieve a signal that simply is not there.

With this understanding in place, we now turn to the second core pillar of effective forecasting: the accurate measurement of forecast performance. This chapter introduces the essential landscape of forecasting metrics, examining their mathematical properties, business relevance, and limitations. From point forecast errors to probabilistic scoring rules and shape-aware alignment measures, we explore how the way we measure forecasts shapes what we build, evaluate, and deploy.

Forecasting performance is only as meaningful as the metrics used to evaluate it. A well-chosen metric aligns closely with business objectives and the nature of the data, whereas a poor metric can mislead model selection and optimization. In this chapter, we survey point forecasting metrics (for single-valued predictions) and probabilistic forecasting metrics (for distributional predictions), clearly separating their uses. We critically examine the strengths and weaknesses of each metric and incorporate insights from forecastability measures (discussed in Chapter 2) – such as entropy-based complexity indices – to understand how the inherent predictability of a time series can guide the selection of metrics.

3.1 Point Forecast Accuracy Metrics (Deterministic Forecasts)

Point forecast metrics assess the error between a single forecast value and the actual outcome. Let y_t be the actual value at time t and $\hat{y}_t$ the forecast. The most common point metrics are based on absolute or squared errors (H is the forecast horizon).

- **Mean Absolute Error (MAE):** $\frac{1}{H}\sum_{t=1}^{H}|y_t - \hat{y}_t|$. The mean absolute error (MAE) measures the average magnitude of the forecast errors without considering direction. It is easy to interpret; for example, an MAE of 5 means that forecasts are off by 5 units on average. MAE treats all errors equally, making it more robust to outliers than squared-error metrics like MSE, which disproportionately penalize larger errors. .
- **Mean Squared Error (MSE):** $\frac{1}{H}\sum_{t=1}^{H}(y_t - \hat{y}_t)^2$. The mean squared error (MSE) is the average of the squared differences between the predictions and the actual values. Squaring the errors ensures that all deviations are positive and emphasizes larger errors. This makes MSE useful when large errors are especially costly, but it is also sensitive to outliers.
- The root mean square error (RMSE) is the square root of the MSE and expresses the error in the same units as the data $RMSE = \sqrt{MSE} = \sqrt{\frac{1}{H}\sum_{t=1}^{H}(y_t - \hat{y}_t)^2}$. RMSE is widely used for its mathematical convenience and sensitivity to large errors. However, like MSE, it is less robust to outliers.

These scale-dependent metrics are intuitive (in units of y) but cannot be used to compare errors across series of different scales. However, it is often recommended to report at least one of such metric for interpretability, and a good practice in many large-scale studies is to report both MAE and RMSE. One drawback is their sensitivity to outliers: a single large

error can dominate the MSE/RMSE. MAE, being linear, is more robust to outliers than MSE. Another drawback is that these raw metrics do not provide context – an RMSE of 10 may be excellent for a volatile series but terrible for a stable series.

3.1.1 Forecast Bias Metric in Point Forecast Evaluation

Forecast bias is a critical aspect of point forecast evaluation. It reflects the tendency of a model to systematically overestimate or underestimate the actual values. The metric commonly used to quantify forecast bias is **Mean Error (ME)**, defined as:

$$\text{ME} = \frac{1}{H}\sum_{t=1}^{H}(y_t - \hat{y}_t)$$

where:

- y_t is the actual observed value at time t,
- $\hat{y}_t$ is the forecasted value at time t,
- H is the number of forecast-observation pairs.

This measure captures the *direction* of the error, which is important when a practitioner needs to detect consistent over-forecasting (negative ME) or under-forecasting (positive ME). A value near zero indicates that the forecasts are unbiased on average. However, ME does not account for the magnitude or variability of errors and should therefore be interpreted alongside scale-independent or absolute error measures such as MAE or MASE.

Metric	Interpretation
Mean Error (ME)	Captures average signed forecast error (bias) Positive ME: under-forecasting; Negative ME: over-forecasting ME near 0: unbiased on average

Table 3.1: Mean Error as a bias metric for point forecast evaluation.

3.1.1.1 Cumulative Forecast Error (CFE)

Cumulative Forecast Error (CFE) measures the total accumulated forecast error over the evaluation horizon:

$$\text{CFE} = \sum_{t=1}^{H}(y_t - \hat{y}_t)$$

CFE is a nonaveraged form of ME and highlights the overall directional error over time. It is especially relevant in applications like inventory management, where cumulative bias can lead to persistent shortages or surpluses.

3.1.1.2 Tracking Signal (TS)

The tracking signal (TS) normalizes the CFE using the Mean Absolute Deviation (MAD) and is used to detect persistent bias.

$$\text{TS} = \frac{\text{CFE}}{\text{MAD}}$$

A tracking signal outside the range of approximately $[-4, +4]$ typically indicates a need to revise the forecasting model, as it suggests a consistent bias that is unlikely due to random variation.

Metric	Formula	Use Case
CFE	$\sum(y_t - \hat{y}_t)$	Total directional error
TS	$\frac{\text{CFE}}{\text{MAD}}$	Normalized bias monitoring

Table 3.2: Common practical extensions for monitoring forecast bias.

3.1.2 Percentage Error Metrics and Pitfalls of MAPE

To compare errors relative to the magnitude of actual values, percentage-based metrics are common. The Mean Absolute Percentage Error (MAPE) is defined as $\frac{100\%}{H}\sum_{t=1}^{H}\left|\frac{y_t-\hat{y}_t}{y_t}\right|$. By expressing errors as a percentage of actuals, MAPE is scale independent and easy to interpret ("on average, the forecast is off by X%"). The easy interpretation makes MAPE very popular. However, its appealing simplicity hides serious defects and risks:

- **Undefined or Extreme Values for Low Demand:** If any actual $y_t = 0$, MAPE is undefined (division by zero). Even very small actual values can blow up the percentage error. For example, forecasting 10 units when the actual is 1 yields an absolute percentage error of 900%. In practice, this makes MAPE unusable for series with zeros or near-zeros – a common occurrence in intermittent demand or certain daily data (e.g., no sales on some days).
- **Asymmetry & Bias Favoring Under-Forecasts:** MAPE puts actual value in the denominator, which means over-forecasting (forecast $\hat{y}_t$ too high) is penalized more severely than under-forecasting by the same amount. In fact, MAPE is asymmetric: if $\hat{y}_t$ is double y_t, that contributes 100% error, but if $\hat{y}_t$ is half of y_t, that's only 50% error. Figure 3.1 illustrates this asymmetry. As a result, optimizing for MAPE tends to produce under-forecasting – the model learns that it is "safer" to under-predict than over-predict. This bias is highly undesirable in many business settings. Goodwin & Lawton [37] formally showed that even the so-called 'symmetric' version of MAPE does not eliminate bias.
- **Aggregation Issues:** A single high-volume series and a low-volume series can have similar percentage errors, but their absolute importance differs. MAPE gives equal weight to each period or series regardless of scale. It "masks performance issues at aggregation" — a large error on a critical high-volume item can be diluted by many small-percentage errors on low-volume items. Without weighting, MAPE cannot differentiate which errors matter more to overall business impact
- **Non-interpretability at Low Averages:** While "10% error" sounds like a clear statement, if that 10% came mostly from periods of tiny actual values, it might be misleading in practical terms of units. Business stakeholders often care about absolute unit errors for impact (e.g. 100 units surplus or short).

In summary, despite MAPE's popularity, there is now a solid consensus among forecasters that MAPE is a poor choice, except in limited conditions. A provocative statement by a leading supply chain analytics firm went as far as: "MAPE has served its duty and should now retire" (Malte Tichy, [95]).

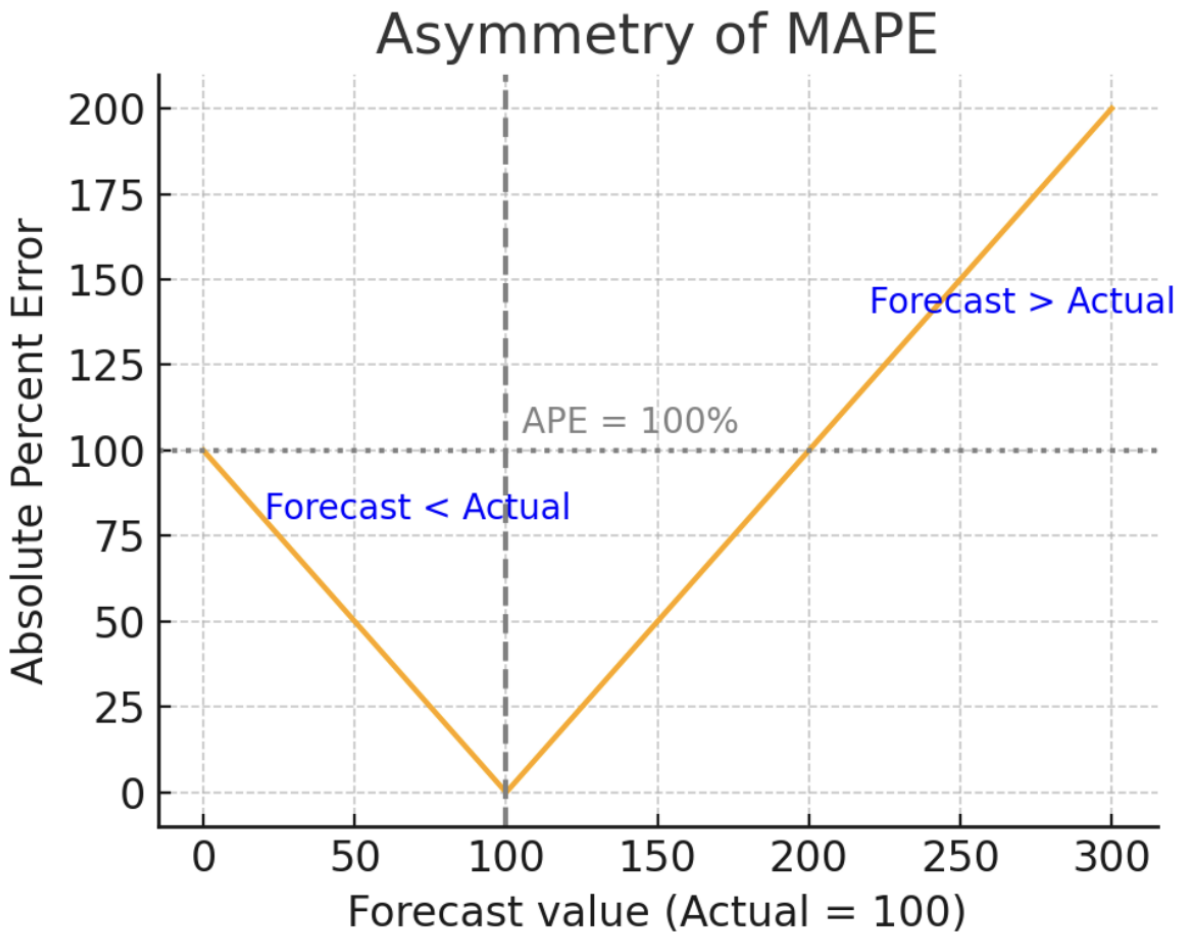

Figure 3.1: Asymmetry of MAPE: MAPE penalizes over-forecasting more heavily than under-forecasting. Equal errors above the actual value result in higher percentage errors than those below, creating a bias that incentivizes under-forecasting. As forecasts increase beyond actual values, the Absolute Percentage Error (APE) can exceed 100% without bound, while forecasts approaching zero cap the APE at 100%. Despite its popularity, this asymmetry is a key limitation of MAPE

Several alternatives address MAPE's issues:

3.1.2.1 Weighted Absolute Percentage Error (WAPE)

Weighted Absolute Percentage Error (WAPE) — also called the MAD / Mean ratio. This is essentially $100\% \times \frac{\sum_{t=1}^{H} |y_t - \hat{y}t|}{\sum_{t=1}^{H} y_t}$. Instead of averaging percentage errors, it divides the total absolute error by the total actual demand. WAPE is volume-weighted, so a 10-unit error on a large-demand item contributes less to the metric than a 10-unit error on a small item.

This directly addresses the aggregation issue – high-volume series dominate the metric as they should. WAPE is also bounded: if any actual $y_t = 0$, the total actual $\sum y_t$ might still be large, so that one zero does not blow up the whole metric. In practice, WAPE behaves similarly to MAPE for series without intermittency or scale extremes but is much more stable in real-world use.

Many businesses prefer WAPE and call it 'MAPE' internally, but it is important to note that

WAPE is a sum-weighted measure (also sometimes called sMAPE in business, though that term conflicts with another metric).

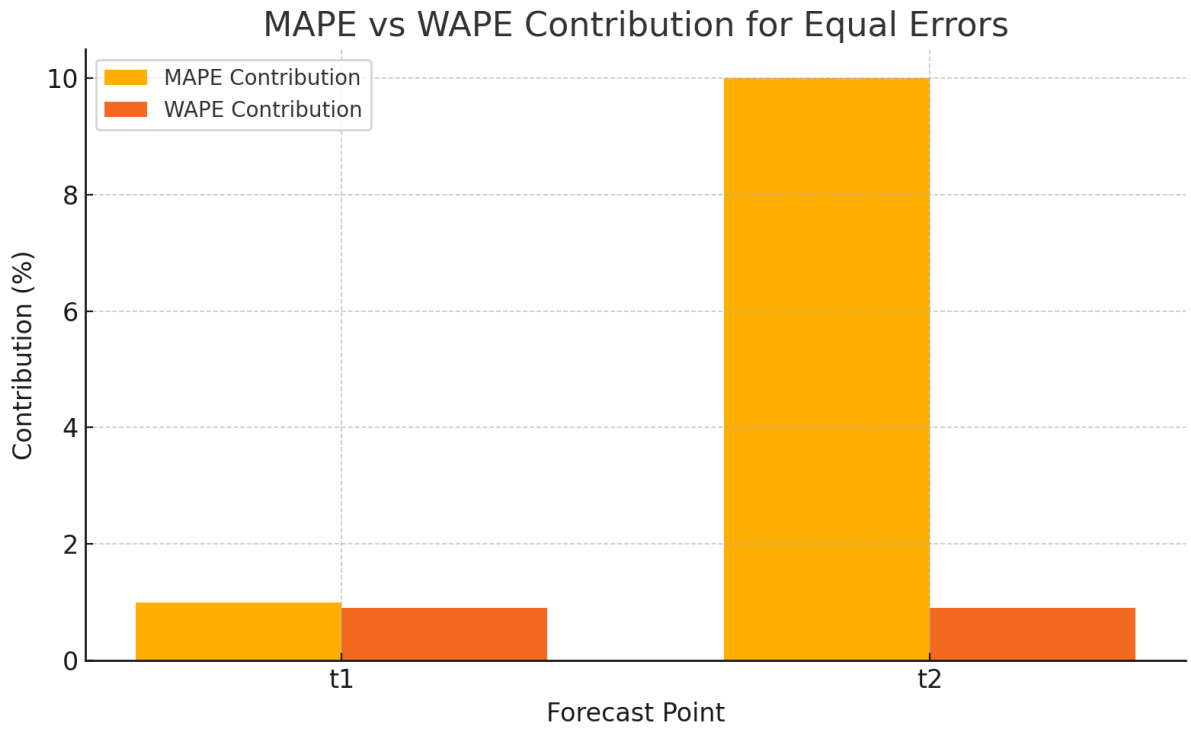

Figure 3.2: MAPE Vs WAPE Contribution For Equal Errors. This chart illustrates how two identical forecast errors (10 units) are evaluated under MAPE and WAPE. MAPE treats both equally, regardless of actual volume. WAPE, by contrast, gives proportionally more weight to errors in high-volume periods. This makes WAPE a more business-aligned, volume-sensitive metric.

A rule of thumb: Use WAPE (MAD/Mean) over MAPE for any practical forecast evaluation, especially in retail or portfolio contexts.

3.1.2.2 Symmetric MAPE (sMAPE)

Symmetric MAPE (sMAPE) is defined as:

$$\text{sMAPE} = \frac{100\%}{T} \sum \frac{|y_t - \hat{y}_t|}{(|y_t| + |\hat{y}_t|)/2}.$$

It uses the average of the forecast and the actual in the denominator to bound the error between 0% and 200%. sMAPE was used as the primary accuracy metric in the M4 Competition (2018).

Although it was theoretically designed to treat under- and over-predictions more'symmetrically' than MAPE, in practice sMAPE introduces its own form of asymmetry. Specifically, it tends to punish *underforecasting* more severely than overforecasting, producing a *positive bias*—the opposite of MAPE's known bias. This arises from the structure of the denominator, which includes the forecast value: when the forecast is lower than the actual, the denominator shrinks, inflating the error; when the forecast is higher, the larger denominator suppresses the error. As shown by Goodwin & Lawton [37], this leads to unequal error penalties for equal-magnitude deviations above and below the actual value. Figure 3.3 illustrates this

asymmetry. Moreover, sMAPE is undefined when both y_t and $\hat{y}_t$ are zero, which can be problematic in zero-heavy time series. Because sMAPE does not fully resolve the limitations of MAPE and introduces new issues with interpretability and bias, it is generally not recommended except for legacy comparisons or where consistency with prior benchmarks is required.

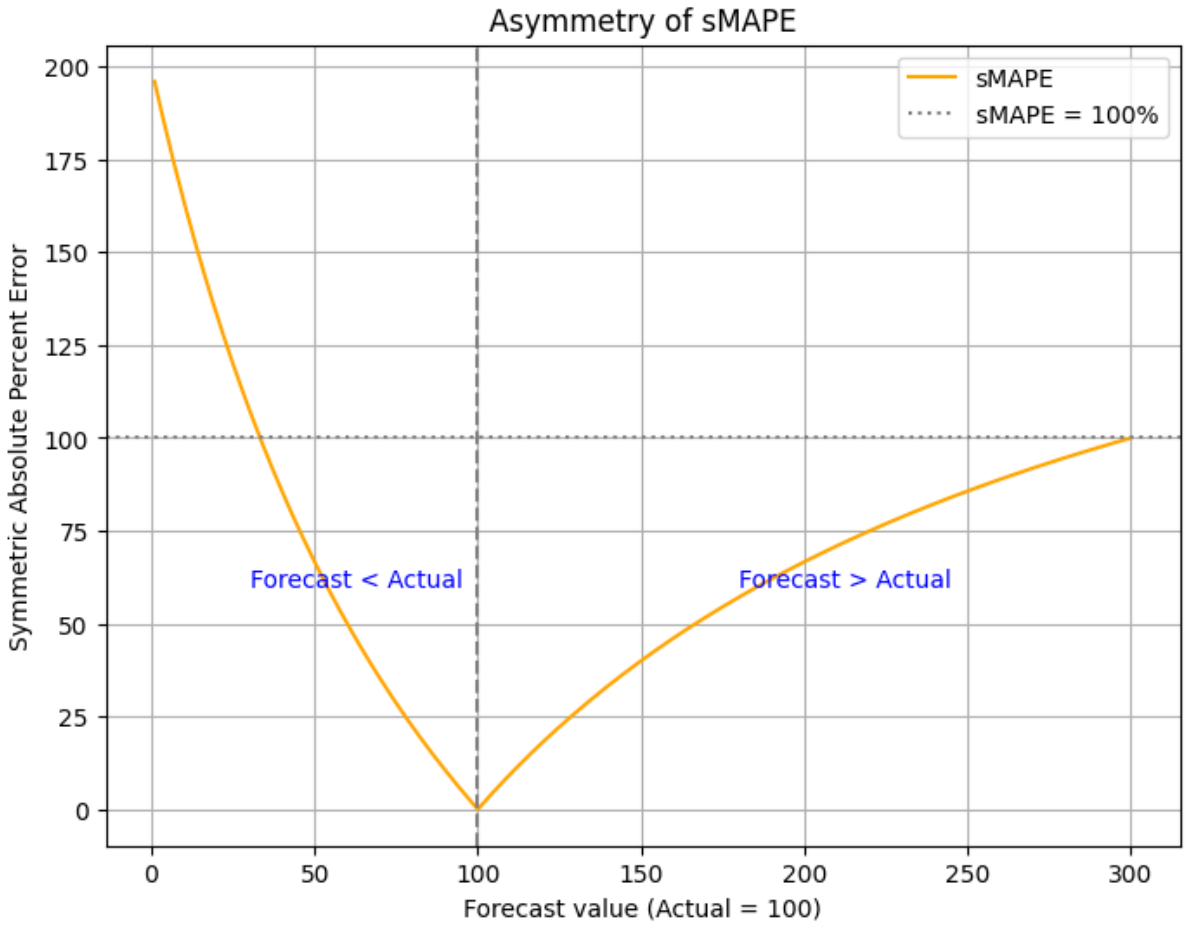

Figure 3.3: Asymmetry of sMAPE: Although designed to address MAPE's bias, sMAPE introduces its own asymmetry. It penalizes under-forecasting more heavily than over-forecasting. For equal errors above and below the actual value, sMAPE produces higher percentage errors when the forecast is below the actual. This occurs because the denominator, which averages the forecast and actual, is smaller when the forecast is low, inflating the error. As forecasts fall below the actual, the Symmetric Absolute Percentage Error (sAPE) rapidly approaches its upper bound of 200%, while over-forecasting leads to more gradual increases. This imbalance incentivizes over-forecasting and highlights a key limitation of sMAPE, as shown by Goodwin & Lawton [37].

3.1.2.3 Mean Arctangent Absolute Percentage Error (MAAPE):

Mean Arctangent Absolute Percentage Error - a less common but clever alternative proposed by Kim & Kim [60]:

$$MAAPE = \frac{1}{H}\sum_{t=1} H \arctan\left(\frac{|y_t - \hat{y}_t|}{|y_t|}\right).$$

By taking the arctangent of the percentage error, it transforms the unbounded $[0, \infty)$ range of MAPE into a $[0, \pi/2]$ range (roughly $[0°, 90°]$ when interpreted as an angle). This naturally constrains extreme errors and avoids divergence at $y = 0$. MAAPE is more robust with intermittent demand and outliers, although its values are expressed in radians (or degrees) rather than intuitive percent. Although promising, MAAPE is not yet widely used in industry; it mainly appears in academic studies of forecasting metrics.

In practice, if percentage errors are desired, using WAPE (MAD/Mean) is the safest general solution. It retains interpretability (for example, "5% of total demand in error") and is never infinite. However, for most cases, the scaled errors described next are even more informative.

3.1.3 Scale-Independent Metrics Errors (MASE, RMSSE)

Instead of percentages, another approach to achieve scale independence is to scale forecast errors by the errors of a benchmark method. The most popular is scaling by a naive forecast. The Mean Absolute Scaled Error (MASE) was introduced by Hyndman & Koehler [54] to address the flaws in MAPE. MASE is defined as:

$$\text{MASE} = \frac{\frac{1}{T}\sum_{t=1}^{T} |y_t - \hat{y}_t|}{\frac{1}{T-\ell}\sum_{t=\ell+1}^{T} |y_t - y_{t-\ell}|},$$

where the denominator is the average one-step naive forecast error in the sample. For nonseasonal data, $\ell = 1$ (last-value naive); for seasonal data, $\ell =$ season length (seasonal naive). The idea is to make an error relative to the way a simple naive method would do. MASE > 1 means the forecast is worse than naive on average;

MASE < 1 indicates that the forecast performs better than the naïve method. It also provides an intuitive scale: for example, a **MASE of 0.8** means that the forecast is **20% better** than naïve, since $1 - 0.8 = 0.2$, or an improvement 20%.

MASE has several advantages: It is well defined for all series (the naive error will only be zero for a constant series, in which case the forecast error is also zero, so one can define MASE = 0 in that trivial case). It also treats over- and under-forecast errors equally (being based on MAE in the numerator) and avoids percentage distortions. It became a de facto standard in forecasting research (e.g., used in M4 competition alongside sMAPE). That said, MASE is not perfect:

1. MASE inherits properties of the naive benchmark. If the naive method is too simplistic (e.g. a non-seasonal naive applied to a seasonal series without adjustment), the denominator may be large, making MASE appear artificially low (too optimistic). It is crucial to use a sensible naive - typically last season value - for seasonal data.
2. MASE does not convey an absolute scale of errors. Business users may not intuitively grasp "MASE = 0.8" as easily as "MAE = 500 units". It is a relative measure.
3. Aggregating MASE across multiple series requires care. Taking a simple average of MASE across series can be misleading, since each series has its own scale. Sometimes a median of MASE values or a weighted average by volume is preferred
4. Since MASE uses MAE, it aligns with optimizing median forecast (MAE is minimized by median). In intermittent demand contexts, the median might be zero even though the mean is higher, leading to a situation where always forecasting zero yields a decent MASE but is clearly a bad business decision. This again underscores that

metric alignment with business goals is vital: if the cost of underforecasting is high, an accuracy metric that rewards zero forecasts can be dangerous.

Despite these caveats, MASE remains a robust, scale-free metric. A complementary metric is Root Mean Squared Scaled Error (RMSSE), which scales the RMSE by the RMSE of a naive benchmark (often using differenced series for scaling). RMSSE was extensively used in M5 forecasting competition for retail demand, where a weighted RMSSE (WRMSSE) was the main evaluation metric. Each series (product-store combination) was assigned a weight proportional to its sales volume, and the RMSSE for each series (with seasonal naive reference) was combined into one weighted score. This ensured that accuracy on high-volume series contributed more to the final score and that scale differences were normalized via the naive benchmark.

In summary, for point forecasts, we recommend reporting at least one scale-dependent metric like MAE or RMSE (in meaningful units) and one scale-independent metric like MASE/RMSSE or WAPE. This gives a balanced view: stakeholders see the typical size of errors, and as analysts, we see whether our model beats simple baselines and by what margin. If using percentage-style metrics, prefer WAPE or at least sMAPE / MAAPE to raw MAPE to avoid the latter's pitfalls.

While accuracy metrics help evaluate absolute error levels, they don't tell us whether a forecast is better than a baseline. This is where relative performance metrics like Forecast Value Added (FVA) come in.

3.2 Forecast Value Added (FVA): Quantifying the impact of the model

In applied forecasting, especially when replacing judgmental or legacy models, it is crucial not just to measure accuracy but to demonstrate value over the current baseline. **Forecast Value Added (FVA)** is a relative metric introduced by Gilliland [34] to exactly quantify this.

Definition

Let $\text{MAE}_{\text{model}}$ and $\text{MAE}_{\text{benchmark}}$ be the mean absolute errors of the model and the baseline (e.g., naive, seasonal naive, or expert forecast). Then:

$$\text{rMAE} = \frac{\text{MAE}_{\text{model}}}{\text{MAE}_{\text{benchmark}}} \quad \Rightarrow \quad \text{FVA} = (1 - \text{rMAE}) \times 100$$

A positive FVA indicates an improvement over the baseline; negative FVA indicates degradation.

Use Cases

- **Retail:** Evaluate whether a statistical model outperforms human planners or seasonal naive forecasts at the SKU or store level.
- **Finance:** Compare a predictive ML model with a random walk or a benchmark from the previous year in revenue forecasting.
- **Energy:** Quantify improvement over historical profiles (e.g., " the same hour yesterday" or climatology) in the load or renewable output forecasts.

Pros and Considerations

- ✓ Intuitive: The stakeholders understand how much better the forecast is.

3.7.2 Aggregation Across Multiple Series

When aggregating errors across N time series, especially with differing scales and importance, several strategies apply:

- **Micro-averaging (volume-weighted)**: Pool all forecasts and compute an overall metric. For MAE:
$$\text{MAE}_{\text{micro}} = \frac{\sum_{i=1}^{N}\sum_{t}|y_{i,t} - \hat{y}_{i,t}|}{\sum_{i=1}^{N} T_i}$$
This approach emphasizes series with more data points or higher volume.
- **Macro-averaging (series-level)**: Compute the error for each series and take the mean:
$$\text{MAE}_{\text{macro}} = \frac{1}{N}\sum_{i=1}^{N}\text{MAE}_i$$
This treats all series equally, regardless of their scale or volume.
- **Weighted averaging**: Use domain-specific weights, such as revenue share or recent sales volume, to reflect business impact.
- **Scaled metrics (e.g., MASE, relative RMSE)**: To allow fair aggregation across series of different scales, use scale-independent metrics such as:
$$\text{MASE}_i = \frac{\text{MAE}_i}{\text{MAE}_{\text{naive},i}}, \quad \text{RelRMSE}_i = \frac{\text{RMSE}_i}{\text{RMSE}_{\text{benchmark},i}}$$
These can be averaged across all series to assess average relative performance.
- **Robust aggregation**: Consider using the median or trimmed mean of the errors per series to mitigate the effect of outliers.
- **Distribution summaries**: Plot histograms or box plots of errors per series to understand the full error profile across the portfolio.

3.7.3 Probabilistic Forecast Aggregation

For distributional forecasts, metrics like CRPS and pinball loss (quantile loss) are aggregated similarly:

- **Mean CRPS or quantile loss** across all forecasted points is the standard.
- **Scaled CRPS**: Normalize by the CRPS of a naïve benchmark for comparability.
- **Coverage-based aggregation**: Compute the overall empirical coverage of prediction intervals across all time points and series.

1.7.4 Summary

To evaluate models across multiple series and horizons:

- Always report both overall and horizon-specific metrics.
- Normalize or weight appropriately when aggregating across heterogeneous series.
- Supplement scalar metrics with distributional and visual summaries.

Improper aggregation can lead to misleading conclusions, particularly when model performance varies across subgroups or horizons. Aggregating thoughtfully ensures metrics are informative, actionable, and aligned with the objectives of the forecast.

3.8 Practical Metric Selection Framework

1. **Understand the Decision Context:** Identify what matters if the forecast is used in decision making. Is under-forecasting or over-forecasting worse (asymmetry)?

Are absolute units more important than % error? Is the timeliness of the prediction critical? For example, in inventory management, stockouts (under-forecast) are worse, so one might incorporate a service level or use a higher quantile forecast; in budgeting, over-forecasting revenue might be worse (overly rosy projections).

2. **Examine Data Characteristics (Forecastability):** Compute basic stats for each series: volatility (CV), intermittency (count of zeros, ADI), seasonality strength, trend, entropy measures. This will tell you which series are easy or hard. Hard series might need special treatment (or at least tempered expectations). Easy series set the bar for performance (if you can't get low error there, the model needs improvement).
3. **Choose a Primary Scale-Dependent Metric:** Typically MAE or RMSE. MAE is often preferable for business interpretability (and robustness), but RMSE if you really care about outliers or have normally distributed errors. This will be used to communicate 'the typical error is... units (or currency)". Ensure to compute this on the same scale as decisions (e.g. if daily errors of 100 look bad, but monthly aggregated error is what matters, maybe focus on monthly MAE).
4. **Choose a Primary Scale-Independent Metric:** Either a percentage (WAPE) or a scaled error (MASE/RMSSE). This is used for comparing across series or against benchmarks. For example, "our model has MASE = 0.8, so 20% better than last year's same period" gives a nice sense of value-add. If using percentage, prefer WAPE to avoid MAPE issues. If you must use MAPE (due to convention), be aware of its issues and perhaps exclude zero-demand periods from the calculation to avoid infinity – or at least report a trimmed MAPE.
5. **If Probabilistic Forecasts Are Relevant:** Include CRPS or an average pinball loss as a metric. For instance, if you generate 100 simulations or a full predictive distribution, report the mean CRPS across the test set. If stakeholders care about certain quantiles (like P90 for risk), report the calibration and pinball loss at P90. Always accompany these with a plain-language interpretation ("the P90 forecast is on average 10 units too low or too high, which is acceptable for our risk tolerance" or "our 95% intervals contained the true value 95% of the time – indicating slight under-dispersion, which we will address").
6. **Consider Specialized Metrics as Needed:** If your use case has specifics like timing (e.g. peak timing in load forecasting) or alignment (e.g. phase of demand surges), add a metric for that. E.g., "Mean timing error of peak demand = 1.5 hours". Or if you have hierarchical structure, ensure a weighted aggregate metric is reported. If forecasts will be reconciled, perhaps coherence isn't a concern; if not, you might at least quantify the incoherence.
7. **Use Multiple Metrics and Visualizations:** No single metric gives the full picture. It's good practice to present a dashboard of metrics – e.g. MAE, MASE, WAPE, bias, and maybe a calibration plot or distribution metric. Ensure none of these tell conflicting stories. If they do (e.g. one model better in MAE, another better in MASE), dig deeper – maybe one model excels on big items, another on small, etc.
8. **Align Metric with Model Optimization if Possible:** If you have the freedom to choose a loss function for training your model, try to match it to the metric. For example, optimizing a neural network on MAE will directly target median and usually yield lower MAE. If you care about MASE, you might incorporate a seasonal naive scale in the loss or use relative error loss. Caution: sometimes you might not optimize on the final metric due to stability (e.g. MAPE is horrible to optimize, so you might optimize MASE or MAE instead, which usually correlates well). Always validate that optimizing your chosen loss actually improves your key metric on a validation set.

9. **Watch out for Gaming and Incentives:** Metrics drive behavior. If forecasters or planning teams are judged by a metric, they may unconsciously game it. For example, a supply planner who is evaluated on MAPE might intentionally under-forecast to avoid high percentage errors on slow movers – but that will cause stockouts (a phenomenon observed in a Blue Yonder simulation where optimizing MAPE alone led to severe stock issues [95]). To counteract this, use multiple metrics (so you can't game one without hurting another) and include ones that capture the undesired side effects (e.g. track bias separately, service level, etc.). In our example, adding a bias metric or a fill rate requirement will prevent the planner from over-focusing on MAPE at the expense of stock.
10. **Iterate Based on Results:** Once you evaluate your model(s), you might find the metric choice needs adjustment. Perhaps every model's MAPE is high due to many zeros – that might push you to switch to WAPE or segment the evaluation. Or you find two models have similar MAE but one has much better P90 error – if P90 is critical, you'd choose that model and realize pinball loss at 0.9 quantile should be a reported metric.

Implementation Tip. For hands-on implementation, we provide a cross-reference table in Appendix 6 linking the core forecasting metrics discussed in this chapter to their corresponding functions in popular Python libraries such as `scikit-learn`, `Nixtla's UtilsForecast`, `scores`, and `properscoring`. This crosswalk enables rapid application of the metrics in code, ensuring that the theoretical insights presented here are directly translatable into practice. Whether you are evaluating probabilistic forecasts with CRPS or benchmarking point accuracy with MAE and MASE, the appendix serves as a practical guide to choosing the right tool for your pipeline.

3.8.1 Nixtla's StatsForecast:

StatsForecast by Nixtla [72] is a high-performance library for statistical time series forecasting (ARIMA, ETS, etc.), often used for large-scale forecasting. Alongside models, Nixtla provides a utility module, "Utilsforecast", for evaluation metrics and other tools. The Nixtla ecosystem (which also includes MLForecast and NeuralForecast) emphasizes efficiency and accuracy and offers solid documentation (Nixtla's 'Nixtlaverse') [93].

Metrics & Features: StatsForecast itself exposes a convenient cross-validation method that can generate historical fold forecasts and compute error metrics such as MAE, MAPE, MASE, RMSE and sMAPE. Under the hood, it relies on Utilsforecast which implements many metrics. Scale-dependent point metrics supported include MAE, MSE/RMSE, and others like MAPE and sMAPE (Symmetric MAPE). The library also provides MASE (Mean Absolute Scaled Error), which requires a seasonal period and training data to compute the scaling term. MASE is computed relative to the naive forecast error in the sample (requiring passing train_df to metrics). This allows for more robust comparisons between series.

For probabilistic forecasts, Nixtla's utilsforecast supports quantile-based metrics. It has a quantile loss (pinball loss) for quantile forecasts, an mqloss for multiquantile average loss, and computes coverage of prediction intervals. A notable advanced metric is the scaled CRPS — a variant of CRPS that normalizes the score by the magnitude of the series (as described by Rangapuram et al. [81]). This provides a single-number metric for the quality of the distribution forecast considering calibration and sharpness.

In addition, calibration and coverage metrics are offered to assess if the predicted intervals have the nominal coverage.

3.8.1.1 Coverage and Calibration Metrics

In addition to point forecast metrics, assessing the reliability of probabilistic forecasts is critical—especially when prediction intervals are involved. The `utilsforecast` library provides two core metrics for evaluating prediction intervals: **coverage** and **calibration**. These help determine whether the forecasted uncertainty is well-aligned with actual outcomes.

Coverage The `coverage` metric measures the empirical frequency with which the predicted intervals contain the observed values. For example, a nominal prediction interval 90% should ideally contain the true value 90% of the time. This metric computes the proportion of actual values that fall within the predicted bounds.

- **Input**: Forecasts with upper and lower bounds at specified coverage levels (e.g., 80%, 95%).
- **Output**: Empirical coverage as a value between 0 and 1.
- **Interpretation**: Higher values closer to the user-specified confidence level indicate better coverage.

Calibration The `calibration` metric measures the absolute difference between the nominal interval level and the actual empirical coverage. Indicates whether the predicted intervals of a model are *well-calibrated* - that is, whether they match the target frequency.

- **Input**: Same as for coverage.
- **Output**: Absolute deviation from nominal coverage level.
- **Interpretation**: A value close to zero implies good calibration.

Table 3.3: Comparison of `coverage` and `calibration` metrics

Metric	What it Measures	Good Value Means
`coverage`	Empirical proportion of observations inside the prediction interval	Close to the nominal level (e.g., 0.90 for a 90% interval)
`calibration`	Absolute difference between empirical and nominal coverage	Close to 0 (indicating perfect calibration)

Example Usage

```python
from utilsforecast.losses import coverage, calibration
from utilsforecast.data import generate_series

# Generate synthetic forecast data with prediction intervals
series = generate_series(n_series=10, n_models=2, level=[80, 95])
models = ['model0', 'model1']

# Compute coverage
coverage_df = coverage(df=series, models=models, level=[80, 95])

# Compute calibration
calibration_df = calibration(df=series, models=models, level=[80, 95])
```

Listing 3.1: Python code for computing coverage and calibration with utilsforecast

These metrics provide a clear quantitative assessment of the way well a model captures forecast uncertainty. Together, they allow practitioners to assess not just the accuracy of point forecasts but also the *reliability* of prediction intervals - vital for risk-sensitive decision-making.

In summary, Nixtla's Utilsforecast suite spans standard errors to sophisticated probabilistic metrics. StatsForecast makes backtesting easy via its cross-validation utility, which iteratively retrains models on expanding windows and computes the chosen metrics on each forecast horizon. This automates the evaluation of the roll origin, producing metrics like MAPE or MASE in multiple splits. The design emphasizes scalability (working with pandas or Polars DataFrames and vectorized operations for speed). The Nixtla libraries are actively maintained, and documentation is available on Nixtla's site with examples [93].

3.9 Advanced and Emerging Metrics

Although traditional metrics such as MAE, RMSE, and CRPS form the backbone of forecast evaluation, a growing body of research and applied forecasting practice has introduced more nuanced and task-specific alternatives. These metrics are particularly valuable in real-world environments where scale, directionality, risk asymmetry, or business cost must be accounted for.

This section presents advanced metrics—**Mean Scaled Interval Score (MSIS)**, **Geometric Mean Relative Absolute Error (GMRAE)**, and **Median MASE**—highlighting their rationale, formulas, and implementation, particularly in multiseries contexts.

3.9.0.1 Mean Scaled Interval Score (MSIS)

The **Mean Scaled Interval Score (MSIS)** is an advanced metric for evaluating the quality of prediction intervals. It accounts for both the width of the interval and whether the true value falls inside it. MSIS generalizes the interval score by scaling it relative to a naive seasonal forecast.

Formula: Given lower and upper prediction bounds ℓ_t and u_t at time t, the MSIS at nominal coverage level α is:

$$\text{MSIS} = \frac{1}{H}\sum_{t=1}^{H}\Big[(u_t-\ell_t)+\frac{2}{\alpha}(\ell_t-y_t)\mathbf{1}\{y_t<\ell_t\}+\frac{2}{\alpha}(y_t-u_t)\mathbf{1}\{y_t>u_t\}\Big]\Big/Q$$

where: - H is the forecast horizon, - y_t is the observed value, - Q is the scaling factor: the in-sample MAE of the seasonal naive forecast.

Interpretation: - A lower MSIS indicates better forecast interval quality. - It penalizes wide intervals and poor coverage asymmetrically. - Scaling by Q makes MSIS unit-free and comparable across series.

```python
def msis(y_true, lower, upper, alpha, naive_errors):
    width = upper - lower
    penalty_lower = (lower - y_true) * (y_true < lower)
    penalty_upper = (y_true - upper) * (y_true > upper)
    interval_score = width + (2 / alpha) * (penalty_lower + penalty_upper)
    scale = np.mean(np.abs(naive_errors))  # seasonal naive MAE
    return np.mean(interval_score) / scale
```

Listing 3.2: Python-style MSIS implementation

Use case: MSIS was used in the M4 competition and is especially useful for evaluating *probabilistic forecasts with intervals* across multiple time series. It combines aspects of accuracy (narrow intervals) and reliability (good coverage), scaled for comparability.

3.9.0.2 Geometric Mean Relative Absolute Error (GMRAE)

GMRAE uses the geometric mean to aggregate relative errors across multiple series:

$$\text{RAE}_{i,t} = \frac{|y_{i,t} - \hat{y}_{i,t}|}{|y_{i,t} - y^*_{i,t}|}, \quad \text{GMRAE} = \left(\prod_{i,t} \text{RAE}_{i,t} \right)^{1/N}$$

Here, $y^*_{i,t}$ is the forecast from a benchmark (e.g., naive). GMRAE < 1 implies the model outperforms the benchmark.

```python
import numpy as np

def gmrae(errors_model, errors_benchmark):
    rae = np.abs(errors_model) / np.abs(errors_benchmark)
    return np.exp(np.mean(np.log(rae)))
```

Listing 3.3: Python-style pseudocode for computing GMRAE

Use case: GMRAE is especially effective for comparing performance across heterogeneous series. Geometric averaging avoids domination by large outliers and satisfies multiplicative fairness.

3.9.0.3 Median MASE (Mean Absolute Scaled Error)

MASE compares forecast error to the in-sample naive forecast error:

$$\text{MASE} = \frac{\frac{1}{H} \sum_{t=1}^{H} |y_t - \hat{y}_t|}{\frac{1}{T-1} \sum_{t=2}^{T} |y_t - y_{t-1}|}$$

Median MASE aggregates per-series MASE scores by taking the median across all series.

```python
def mase(y_true, y_pred, y_train):
    scale = np.mean(np.abs(np.diff(y_train)))
    mae = np.mean(np.abs(y_true - y_pred))
    return mae / scale

def median_mase(all_series_forecasts):
    mase_scores = []
    for y_train, y_true, y_pred in all_series_forecasts:
        mase_scores.append(mase(y_true, y_pred, y_train))
    return np.median(mase_scores)
```

Listing 3.4: Python-style pseudocode for computing Median MASE

Use case: Median MASE is a robust measure in multi-series settings, resistant to outliers and interpretable: a value < 1 indicates better-than-naive performance in at least half the series.

3.9.0.4 Summary

Table 3.4: Comparison of Advanced Forecasting Metrics

Metric	Relative	Aggregation	Best for
NSE	No	Over time	Single-series models; hydrology
GMRAE	Yes	Geometric over series or time	Multi-series benchmarking with robustness
Median MASE	Yes	Median over series	Typical-case performance across heterogeneous series

Each of these metrics offers unique strengths. GMRAE and Median MASE are especially appropriate when comparing performance across multiple series, where arithmetic means may be skewed by heterogeneity or outliers. Their mathematical robustness supports more reliable model comparisons in large-scale forecasting tasks.

Skill Scores and Relative Accuracy

- **Theil's U statistic:** A scale-independent benchmark metric that compares forecast performance to a naive forecast or random walk. $U < 1$ implies the model outperforms the benchmark.
- **Nash–Sutcliffe Efficiency (NSE):** Common in hydrology and energy forecasting, measuring the proportion of variance explained compared to the mean.

Geometric Aggregates for Fair Comparison

- **GMRAE (Geometric Mean Relative Absolute Error):** Avoids the distortion introduced by arithmetic means when aggregating ratios across heterogeneous series [25].
- **Median MASE:** Reduces the influence of extreme outliers in multiseries comparisons.

Directional and Classification-Oriented Metrics

- **Directional Accuracy:** Measures how often the forecast correctly predicts the direction of change (e.g. $P(y_t - y_{t-1} > 0) = P(\hat{y}_t - \hat{y}_{t-1} > 0)$). Especially useful in financial or tactical settings.
- **Precision, Recall, and F1 on Threshold Forecasts:** For binary event forecasting (e.g., demand spike or grid overload), classification-style metrics give meaningful insights.

Service-Level and Business Impact Metrics

- **Fill Rate:** Fraction of periods in which the forecast inventory met the actual demand. Highly actionable in inventory planning.
- **Cost-Based Metrics:** Domain-specific loss functions converting forecast errors into economic cost (e.g., underforecasting penalty × lost sales + overforecasting × holding cost).

Takeaway

These advanced metrics often align more directly with business goals and user-facing performance. While they are not universally applicable, they should be part of the forecaster's toolkit, especially when deploying models in production where accuracy must translate into actionable performance.

3.10 Metric Trait Reference: Selecting Metrics by Properties

Choosing a forecasting metric requires more than statistical intuition - it must align with the characteristics of the data, the goals of modeling and the risks of the business. While the previous sections covered individual metrics in depth, this summary matrix synthesizes their comparative traits.

Table 3.8 presents a high-level checklist of properties across key metrics, helping practitioners quickly identify:

- Which metrics are robust to low or zero values
- Which are scale-independent or bounded
- Whether a metric favors mean or median-based predictions
- Whether a metric is a proper scoring rule (incentivizing honest uncertainty estimation)
- Which metrics can safely handle intermittent demand

Metric	Zeros	Scale-Free	Bounded	Proper	Target	Intermittent
MAE	✅	❌	❌	❌	Median	⚠️ Acceptable
RMSE	✅	❌	❌	❌	Mean	⚠️ Sensitive
MAPE	❌	✅	❌	❌	Mean	❌ Poor
WAPE	✅	✅	❌	❌	Mean	✅ Preferred
sMAPE	⚠️ Mixed	✅	✅	❌	Mean	⚠️ Bias Risk
MAAPE	✅	✅	✅	❌	Mean	✅ Good
MASE	✅	✅	❌	❌	Median	⚠️ Depends
RMSSE	✅	✅	❌	❌	Mean	⚠️ Contextual
CRPS	✅	✅	✅	✅	Dist'n	✅ Ideal
Pinball Loss	✅	✅	✅	✅	Quantile	✅ Ideal
Winkler Score	✅	✅	✅	✅	Interval	✅ Strong

Figure 3.8: The matrix summarizes the key properties of common forecast metrics, helping practitioners quickly identify which are suitable based on data characteristics and modeling objectives.

This compact view enables modelers and business stakeholders to select metrics that are not just statistically valid but also *fit for purpose* — resilient, interpretable, and operationally aligned.

3.11 Quick Metric Selection Guide

In practice, choosing the right forecast accuracy metric depends on the characteristics of your time series, the business context, and the type of model output (point vs. probabilistic). The table below serves as a concise decision aid to help practitioners select appropriate metrics based on common forecast scenarios.

If your data or use case has...	Then consider using...	Why?
Forecast errors in raw units matter	MAE, RMSE	Easy to interpret
Large errors are especially undesirable	RMSE, RMSSE	Heavily penalizes outliers
Different series have different scales	MASE, WAPE, RMSSE	Scale-free comparisons
Many zeros or small values	WAPE, MAAPE, MASE	MAPE breaks on zeros
Need to compare against a naive or baseline model	MASE, FVA, Theil's U	Relative accuracy
Probabilistic forecasts (e.g., intervals, distributions)	CRPS, Pinball Loss, Winkler Score	Proper scoring rules
Business impact from under/over-forecasting is asymmetric	Quantile Loss (e.g., P90, P10)	Customizes penalty by quantile
Demand is intermittent or lumpy	MAAPE, WAPE, Service Level	Robust to gaps and erratic values
Forecast timing or shape matters	DTW, DILATE, TDI	Captures event timing and profile similarity
Hierarchical data structure	WRMSSE, Coherence Measures	Accounts for aggregation consistency
Extreme risks or safety-critical systems	CRPS, Interval Coverage, Tail Quantile Loss	Focus on uncertainty and tail accuracy

Table 3.5: Quick reference for selecting forecasting metrics based on data traits and decision context.

3.12 Chapter Summary

In this chapter, we explore a wide range of forecasting metrics and their proper use:

1. **Point forecast metrics** such as MAE and RMSE measure typical errors in units of the forecast variable, while scale-free metrics such as MASE, RMSSE, and WAPE allow comparison between series and against benchmarks. We stressed caution against MAPE due to its bias and instability, suggesting alternatives (WAPE, sMAPE, MAAPE) that mitigate those issues.
2. **Probabilistic metrics** such as CRPS and pinball loss evaluate the quality of forecast uncertainty estimates, ensuring that models not only hit the target on average but also assess their confidence correctly. We introduced the Winkler score for interval forecasts and highlighted the importance of proper scoring rules for comparing probabilistic forecasts on a sound footing
3. **Temporal shape metrics** (DTW, DILATE, TDI, etc.) were discussed as advanced tools for special situations where forecast timing and shape matter more than exact values. These can be valuable for training modern deep learning models and diagnosing performance on events/patterns, though they complement rather than replace traditional metrics.
4. **Hierarchical metrics** such as WRMSSE illustrate how to evaluate forecasts at aggregate levels, using weighting and scaling to ensure fairness and relevance at each level.

A. ARIMA Modeling: Theory, Practice, and Advanced Techniques

> *"Before teaching your model to predict the future, learn how well it understands the past."*
>
> — ***Forecasting Proverb***

Chapter **??** emphasized the crucial role of robust performance evaluation in time series forecasting. We detailed best practices for constructing reliable and reproducible evaluation frameworks, highlighting methods such as time-sensitive cross-validation, rolling-origin evaluation, blocked resampling, and realistic backtesting procedures that closely mirror practical forecasting environments. In this chapter, we shift our focus from evaluating forecasts to examining the modeling techniques themselves. We begin our exploration with classical forecasting methods, specifically addressing two foundational and widely-utilized approaches: Auto-Regressive Integrated Moving Average (ARIMA) models and Exponential Smoothing (ETS) methods.

ARIMA models, grounded in statistical theory, represent a versatile approach to capturing and modeling temporal dependencies within time-series data. Building upon foundational insights from Yule, Slutsky, and Wold, the seminal work by Box and Jenkins synthesized earlier approaches into a comprehensive framework known as the ARIMA model. ARIMA systematically integrates autoregressive processes (AR), differencing to achieve stationarity (the integrated or I component), and moving average (MA) elements to effectively model diverse time-series dynamics, including trends, cyclical behaviors, and stochastic variations.

Indeed, while various classical time-series models might seem like a 'grab bag' of different types, ARIMA models offer a more general class that can encompass many of these. They provide a systematic framework for forecasting any given time series by understanding its underlying statistical properties and dependencies. This systematic approach, formalized by Box and Jenkins, allows a comprehensive methodology for identification, fitting, and diagnostic checking, ultimately leading to robust forecasts.

In this chapter, we explore the ARIMA modeling methodology in depth, covering its historical and theoretical underpinnings, practical implementations, methods for parameter estimation, and key considerations in achieving stationarity and invertibility, crucial for reliable forecasting.

Exponential smoothing methods provide an intuitive yet powerful alternative to ARIMA, known for their simplicity, interpretability, and flexibility in capturing trends and seasonality. Initially popularized by Brown and Holt-Winters, these models have evolved to encompass a comprehensive family capable of handling additive or multiplicative trends and seasonal patterns. This chapter will systematically dissect the theoretical underpinnings of exponential smoothing, including single, double, and triple exponential smoothing approaches, their assumptions, smoothing parameters estimation, and adaptive techniques to accommodate structural changes in data.

ARIMA models can be seen as generalized random walk models fine-tuned to eliminate all residual autocorrelation, and also as generalized exponential smoothing models that can incorporate long-term trends and seasonality. They are arguably the most general class of forecasting models for time series that can be stationarized by transformations such as differencing, logging, or deflating.

Together, ARIMA and exponential smoothing represent the foundational pillars of classical time series forecasting. They not only offer robust baseline models, but also provide critical insights into the structure and predictability of the underlying data, setting the stage for more advanced modeling techniques covered in subsequent chapters.

A.1 Introduction

Time-series forecasting is a critical task across finance, economics, retail, and many other fields. Classical statistical models have long provided robust and interpretable approaches for forecasting. In particular, Autoregressive Integrated Moving Average (ARIMA) models and exponential smoothing methods (including simple exponential smoothing, Holt's linear method, Holt-Winters seasonal method and the ETS framework) are two of the most widely used and enduring approaches [**tanthiamhuat2023forecasting**]. These methods represent complementary philosophies: ARIMA models seek to explain the autocorrelation structure of the data, while exponential smoothing focuses on capturing the trend and seasonal patterns in the data [**tanthiamhuat2023forecasting**].

Despite the emergence of advanced machine learning and algorithms, ARIMA and exponential smoothing remain fundamental due to their theoretical rigor, interpretability, and strong forecast performance in many settings. This chapter provides an in-depth exploration of classical time series models with a focus on ARIMA and exponential smoothing. We will cover the intuition and mathematical formulations of ARIMA (including its special cases AR, MA, ARMA, ARIMA, and seasonal ARIMA) and of exponential smoothing methods (including simple exponential smoothing, Holt's linear trend method, Holt-Winters seasonal methods, and the state-of-the-art ETS framework). We discuss the historical development of these models, their underlying assumptions, and how they have evolved over time.

Practical implementation is highlighted with examples using real-world data. We also introduce multivariate extensions like Vector Autoregression (VAR) for handling multiple time series, and state-space models, which provide a unifying theoretical framework for time series (and underlie the ETS approach). Throughout, Python code snippets illustrate how to implement these models using libraries.

The goal is to blend theory with practice, giving a comprehensive understanding of both how these models work and how to apply them to real data. By the end of this chapter, the reader should have a strong grasp of: (a) the intuition and mathematics behind ARIMA and exponential smoothing models; (b) when and why to use each approach; (c) how to implement them in Python on real datasets; and (d) extensions to more complex scenarios (seasonality, multiple time series, etc.).

We also cite foundational literature – including Box & Jenkins' work on ARIMA and the seminal work of Brown, Holt, Winters and Pegels on exponential smoothing, as well as modern sources like Hyndman & Athanasopoulos's *Forecasting: Principles and Practice*. With a firm grounding in these classical models, one can build a solid foundation for more advanced forecasting techniques.

A.2 ARIMA Models and the Box-Jenkins Methodology

As discussed in Chapter 1 The theoretical foundations of ARIMA modeling trace back to pivotal early statistical contributions in time series analysis. George Udny Yule, in his landmark 1927 paper, laid essential groundwork by demonstrating how autoregressive (AR) processes, where current values of a series depend linearly on its past values, could generate realistic cyclic patterns observed in economic data.

Complementing Yule's findings, Eugen Slutsky independently showed in the same period that applying moving averages (MA), which are linear combinations of past random shocks, could similarly produce cyclical fluctuations, even when starting from purely random data.

Herman Wold further strengthened these concepts in 1938 by proving the decomposition theorem, illustrating that any stationary time series could be expressed uniquely as an infinite moving average of past innovations. Collectively, these seminal ideas provided the theoretical scaffolding upon which modern ARIMA models would eventually be built.

Contrary to common perception, ARIMA did not emerge fully formed in the 1970s. Rather, it synthesized these earlier innovations into a comprehensive and practical methodology. George Box and Gwilym Jenkins, in their influential 1970 book, "Time Series Analysis: Forecasting and Control," [12] popularized the systematic modeling framework now known as the Box-Jenkins methodology. This approach clearly articulated a structured procedure for identifying, fitting, diagnosing, and forecasting ARIMA models, thereby transforming scattered theoretical insights into an actionable forecasting toolkit widely adopted in academia and industry.

A.2.1 Autoregressive (AR) and Moving Average (MA) Components

An **autoregressive** (AR) model specifies that the current value of a time series can be explained by its own previous values. Formally, an AR model of order p, denoted AR(p), is written as:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + \varepsilon_t,$$

where X_t is the value of the series at time t, $\phi_1, \ldots, \phi_p$ are the model parameters, and ε_t is a random error (noise) term. In an AR(p), the intuition is that the series has memory of its previous values p - for example, an AR(1) means X_t depends linearly on just the last value X_{t-1}. AR models were among the first stochastic time series models studied; notably, Yule (1927) used an AR(2) model to analyze sunspot data. This pioneering work introduced the idea of explaining cycles in time series via autoregression, and it was later extended by Walker (1931) and others who developed the Yule-Walker equations for estimating AR parameters (see chapter 1 for more details).

A **moving average** (MA) model, by contrast, expresses X_t as a linear combination of past error terms (shocks or innovations). An MA model of order q, denoted MA(q), is:

$$X_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

where μ is the mean of the series (which can be taken as 0 after de-meaning or differencing), $\theta_1, \ldots, \theta_q$ are parameters, and ε_t again is white-noise error. In an MA(q), the series is a weighted sum of the last q random shocks. The term “moving average” can be misleading here – it does not mean a rolling mean of past observations, but rather a rolling sum of past errors. The effect of a shock ε_{t-k} persists for k periods in an MA(q) model. Early work by Slutzky (1927) and Wold (1938) explored moving average representations of economic time series. In fact, Wold’s Theorem in 1938 established that any stationary time series can be decomposed into an (infinite) moving average of past shocks plus a deterministic part – providing a theoretical justification for using AR and MA components to model time series. Autoregressive Moving Average (ARMA) models: In practice, a pure AR or pure MA model may not be sufficient for many series. The ARMA(p, q) model combines both AR(p) and MA(q) components:

$$X_t = \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q},$$

This parsimonious model can capture a wide range of autocorrelation patterns with relatively few parameters. The AR part accounts for persistence in the levels of the series, while the MA part accounts for short-term shock effects. ARMA models assume that the series is stationary (its statistical properties do not change over time). If the original series is nonstationary (e.g., has a trend or changing variance), we may first differencing it to achieve stationarity, which leads to the 'integrated' part of ARIMA.

A.2.2 Differencing and the Integrated (I) Component

A time series is said to be *stationary* if its statistical properties—such as mean, variance, and autocorrelation—remain constant over time. In other words, the process has no trend, no changing variance (heteroskedasticity), and exhibits a consistent level of fluctuation (or "wiggliness").

However, many real-world time series exhibit nonstationary behavior. This includes patterns like trends, drift, or random walk characteristics. Before applying an ARIMA model, it is often necessary to transform such series into a stationary form.

Differencing is a fundamental technique used to achieve stationarity. It involves computing the differences between observations to remove trends and stochastic components.

- The *first difference* of a series is defined as:

$$\nabla X_t = X_t - X_{t-1}$$

  This operation removes linear trends or random walk behavior.
- A *seasonal difference* of period m is:

$$\nabla_m X_t = X_t - X_{t-m}$$

  This helps eliminate repeating seasonal effects every m time periods.

The **"I"** in `ARIMA` stands for *Integrated*, referring to a series that becomes stationary only after differencing. A time series that must be differenced d times to become stationary is called an *integrated series of order* d. This transformation effectively resolves the *unit root* problem, where shocks to the system result in permanent effects and induce a stochastic trend.

By differencing, we model the changes in the series rather than the level itself. These changes often exhibit more stable statistical properties. For instance, consider a random walk:

$$X_t = X_{t-1} + \epsilon_t$$

Its first difference is:

$$\nabla X_t = X_t - X_{t-1} = \epsilon_t$$

which is simply white noise—i.e., a stationary process with zero mean and constant variance.

The differencing operation can also be expressed using the *backshift operator* B, defined by $BX_t = X_{t-1}$. Using this notation:

$$\nabla X_t = (1-B)X_t \quad \text{and} \quad \nabla^2 X_t = (1-B)^2 X_t$$

This shows how differencing removes deterministic trends by transforming the series into its first or second differences, effectively modeling the local rate of change or acceleration.

In an ARIMA(p,d,q) model, the integrated component reflects the application of $(1-B)^d$ to the series before fitting an ARMA(p,q) model. The value of d is typically selected through statistical tests such as the Augmented Dickey-Fuller (ADF) or KPSS test, or by visual inspection of the series and its autocorrelation structure.

A.2.3 Understanding ARIMA (p,d,q) Models

An **ARIMA**(p,d,q) model is essentially an **ARMA**(p,q) model applied after differencing the series d times. The differenced series is then "integrated" back by accumulating these differences to generate forecasts. Simply put, ARIMA generalizes ARMA models by explicitly accommodating nonstationary data through differencing steps:

- If $d=1$, the ARIMA model first differencing (∇X_t).
- If $d=2$, it involves second-order differencing, etc.

Typically, the number of non-seasonal differences (d) will be 0, 1, or 2. Similarly, the number of seasonal differences (D) should generally not be greater than 1. The sum of non-seasonal and seasonal differencing orders (d+D) should ideally not exceed 2, as higher orders can introduce unnecessary complexity and potential 'overdifferencing' issues.

Including the differencing parameter d significantly expands the ARMA framework, making it suitable for a broader variety of real-world series, particularly those with clear trends.

Examples:

- **Random Walk:** A random walk is inherently nonstationary, but its first differences are stationary (white noise). Thus, it can be represented as an ARIMA$(0,1,0)$ model with a constant (drift).
- An ARIMA(0,2,0) model (without a constant) corresponds to a randomly varying local trend, where the second difference is white noise. This is often used for data whose local trend is stochastic.

International Airline Passengers Data (1949–1960): This classic data set exhibits both a pronounced upward trend and clear seasonal cycles. To achieve stationarity and effectively model it with ARIMA, one might:

- Take a first difference $(d=1)$ to handle the trend.
- Apply seasonal differencing $(D=1$ with period $s=12)$ to manage seasonality.

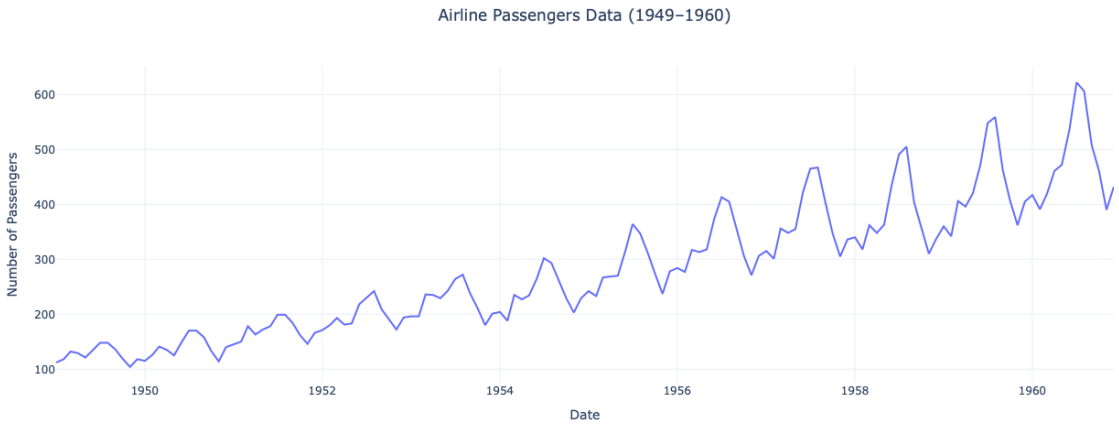

Figure A.1: Monthly airline passengers dataset (1949–1960).This classic time series shows the monthly totals of international airline passengers, highlighting a clear upward trend and pronounced seasonal fluctuations. Originally analyzed by Box and Jenkins in their seminal book "Time Series Analysis: Forecasting and Control" (1970), the dataset is frequently used to illustrate foundational concepts and methodologies in ARIMA modeling and forecasting.

A.2.3.1 The Role of the Drift Term

In ARIMA models, especially when differencing is applied, the drift term acts much like a constant growth rate or slope. It functions analogously to an intercept, but its effect accumulates over time after differencing has been performed. Essentially, it accounts for a consistent, deterministic upward or downward trend in the series that persists even after accounting for past values and random shocks. When differencing reverses the non-stationarity, this drift term becomes a predictable linear increase (or decrease) in the expected value of the original series over time.

Mathematically, consider a simple Random Walk with Drift, which is an ARIMA$(0,1,0)$ model with a drift term:

$$X_t = X_{t-1} + \delta + \epsilon_t$$

where δ is the constant drift term and ϵ_t is white noise. If we repeatedly substitute, we can see its effect on the original series X_t:

$$X_t = X_0 + t\delta + \sum_{i=1}^{t} \epsilon_i$$

Taking the expectation (average value) of this equation:

$$\mathbb{E}[X_t] = X_0 + t\delta$$

This clearly shows that the drift term δ introduces a **deterministic linear trend** to the expected value of X_t. Without the drift ($\delta = 0$), the expected value of a differenced series would remain constant, meaning the original series would exhibit a stochastic trend without a consistent direction.

Deterministic vs Stochastic Trends.

- A **deterministic trend** arises from the drift term. It implies that the trend is predictable and linear over time.

- A **stochastic trend** arises from the accumulation of random shocks (the integrated white noise). It reflects uncertainty in the long-run path of the series.

When modeling real-world time series:

- Include a **drift** if the series shows a consistent long-term direction even after differencing.
- Prefer **stochastic trends** (i.e., no drift) when long-term movements are driven by unpredictable innovations.

In Practice. Software like `forecast::auto.arima()` or `pmdarima` includes options to test whether a drift improves model fit (often via AICc). Including drift in an ARIMA model is especially helpful for economic series like GDP or stock indices, where the cumulative effect of a small trend is meaningful.

Note: Drift is only defined when $d \geq 1$; for stationary models ($d = 0$), the intercept plays a similar role.

Pro Tip: Drift and Long-Term Forecasts The inclusion or exclusion of a drift term significantly impacts long-term forecasts from ARIMA models. A drift term will cause long-term forecasts of a differenced series to follow a straight line with a constant slope. For an undifferenced series, this translates to a linear trend. Omitting a necessary drift can lead to underprediction (or overprediction) if there is consistent underlying growth, while including an unnecessary drift can introduce an artificial trend into your long-term forecasts.

A.2.4 The Box-Jenkins Methodology

The Box-Jenkins methodology: Box and Jenkins (1970) revolutionized time series forecasting by formalizing a three-step iterative process for ARIMA modeling:

A.2.4.1 Identification - Comprehensive Guide to ACF and PACF for ARIMA Model Identification

Use plots of the data, along with its autocorrelation function (ACF) and partial autocorrelation function (PACF), to assess whether differencing is required to achieve stationarity and to help identify initial estimates for parameters p and q. The process of identifying appropriate orders for AR and MA terms relies heavily on analyzing the specific patterns and characteristic'signatures' these functions exhibit for a stationarized series.

For instance, if the ACF decays slowly and remains significantly positive for many lags, this typically indicates non-stationarity (e.g., presence of a trend), necessitating differencing. Seasonal patterns identified through spikes in the ACF/PACF at multiples of the seasonal period suggest that a seasonal ARIMA (SARIMA) model may be suitable.

Stationarity can also be formally assessed using statistical tests such as the augmented Dickey-Fuller (ADF) test. To further clarify the practical use of ACF and PACF in ARIMA model selection, this section offers a comprehensive visual and theoretical guide, highlighting common patterns and their implications for model order specification.

Introduction to ACF and PACF

The **Autocorrelation Function (ACF)** and **Partial Autocorrelation Function (PACF)** are foundational tools in time series analysis. They help diagnose temporal dependencies and guide the selection of models in the Box-Jenkins methodology for ARIMA modeling. These plots visually indicate the nature and extent of correlations across lags in a univariate time

series, supporting effective identification of AR (Autoregressive) and MA (Moving Average) components.

Theoretical Foundations

Autocorrelation Function (ACF)

The ACF measures the linear correlation between observations at different time lags:

$$\text{ACF}(k) = \frac{\text{Cov}(y_t, y_{t-k})}{\sqrt{\text{Var}(y_t)\text{Var}(y_{t-k})}}$$

High autocorrelation at lag k suggests that past values retain predictive information, which is especially useful when identifying MA terms in an ARIMA model.

Partial Autocorrelation Function (PACF)

PACF measures the correlation between y_t and y_{t-k} after removing the effects of intermediate lags:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_k y_{t-k} + \epsilon_t$$

The coefficient ϕ_k in this regression corresponds to the PACF at lag k. A significant spike at lag k implies a direct relationship not mediated by shorter lags.

Typical ACF and PACF Patterns

- **AR(p)**: An AR(p) process shows a PACF that cuts off sharply after lag p. Its ACF, on the other hand, dies out gradually, often with positive autocorrelation at lag 1. This is an 'AR signature'.
- **MA(q)**: MA(q) An MA(q) process has an ACF that cuts off sharply after lag q. Its PACF, by contrast, dies out more gradually, often with negative autocorrelation at lag 1. This is an 'MA signature.'
- In **ARMA(p,q)**, both ACF and PACF typically decay gradually without sharp cutoffs. This mixed behavior makes identification less straightforward. For nonseasonal models, it is often advisable to favor simpler pure AR or MA models unless the data clearly indicate a need for a combined approach.

Illustrative Examples

Example: Airline Passengers Dataset This dataset (1949–1960) exhibits strong seasonal behavior. The ACF shows significant spikes at lags 12, 24, etc., while the PACF plot cuts off after lag 1 or 2, suggesting a SARIMA$(p,d,q)(P,D,Q)_{12}$ model.

A.2.5 Python Implementation Examples

A.2.5.1 Using `statsmodels`

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

data = pd.read_csv(
    'https://raw.githubusercontent.com/valeman/Mastering-Modern-Time-Series-Forecasting-T
    parse_dates=['Month'], index_col='Month'
```

```
)

fig, axes = plt.subplots(1, 2, figsize=(15, 5))
plot_acf(data['Passengers'], lags=30, ax=axes[0])
plot_pacf(data['Passengers'], lags=30, ax=axes[1])
plt.show()
```

A.2.5.2 Using StatsForecast

```
from statsforecast.utils import plot_series
from statsforecast import StatsForecast
from statsforecast.models import AutoARIMA

data = pd.read_csv(
    'https://raw.githubusercontent.com/valeman/Mastering-Modern-Time-Series-Forecasting-T
    parse_dates=['Month']
)

aa = AutoARIMA(season_length=12)
sf = StatsForecast(models=[aa], freq='M', n_jobs=-1)
sf.fit(df=data.reset_index(), id_col='Month', target_col='Passengers')
sf.plot_acf('Passengers', max_lags=24)
sf.plot_pacf('Passengers', max_lags=24)
```

A.2.5.3 Using ETNA

```
from etna.datasets import TSDataset
from etna.analysis import plot_acf, plot_pacf
import pandas as pd

# Load and rename dataset to match ETNA format
data = pd.read_csv(
    'https://raw.githubusercontent.com/valeman/Mastering-Modern-Time-Series-Forecasting-T
    parse_dates=['Month']
)
data = data.rename(columns={'Month': 'timestamp', 'Passengers': 'target'})

# Convert to ETNA-compatible format
ts = TSDataset.to_dataset(data)
ts_dataset = TSDataset(ts, freq='M')

# Plot ACF and PACF
plot_acf(ts_dataset, lags=24)
plot_pacf(ts_dataset, lags=24)
```

Rules of Thumb for Identifying Nonseasonal AR/MA Orders (from Robert Nau)

Use ACF and PACF plots to identify AR and MA terms in nonseasonal data:

- **MA(q) Signature:** If the **ACF cuts off sharply at lag** k (i.e., significant at k, then near-zero beyond), and the **PACF decays gradually**, then set $q = k$ and $p = 0$.
- **AR(p) Signature:** If the **PACF cuts off sharply at lag** k, and the **ACF decays gradually**, then set $p = k$ and $q = 0$.
- **AR(1) vs MA(1):** If there is a **single spike at lag 1** in both ACF and PACF:
  - If positive, set $p = 1$ and $q = 0$ (AR(1) signature).
  - If negative, set $p = 0$ and $q = 1$ (MA(1) signature).

Guidelines for Model Parsimony:

- You should virtually never need $p > 3$ or $q > 3$ in a business application.
- Typically, $p + q \leq 3$, and usually only one of them is non-zero.

Important Note: Do not pay attention to isolated spikes in the ACF or PACF plot beyond lag 3 if you are working with nonseasonal data.

A.2.6 Using ACF/PACF in Box-Jenkins Methodology

1. **Identification**: Use ACF and PACF to suggest model orders. ACF cutoff $\rightarrow$ MA(q); PACF cutoff $\rightarrow$ AR(p).
2. **Estimation**: Fit ARIMA(p, d, q) with selected orders.
3. **Diagnostics**: Examine residual ACF/PACF plots for remaining structure; if residuals resemble white noise, the model is adequate.

Confidence Intervals and Overfitting

ACF and PACF plots include 95% confidence bounds. Spikes outside these bands indicate statistically significant lags. Overfitting can be suspected when:

- Many small spikes appear near the confidence bounds.
- Higher-order lags are included with no clear pattern.
- Residual ACF shows lingering autocorrelation.

Common Pitfalls

- **Non-stationarity**: Long decay in ACF indicates a need for differencing.
- **Seasonality**: Regular spikes in ACF at fixed intervals (e.g., lag 12) require seasonal modeling.
- **Short Series**: Wider confidence intervals make identifying significant lags unreliable.

A.2.7 Advanced Considerations

- **High Noise**: Random noise masks true dependencies; models should be conservatively chosen.
- **Low Noise**: Makes patterns in ACF/PACF more obvious but may tempt overfitting.
- **Long Series**: Provides more accurate estimation of autocorrelations.

Recommended Datasets for Practice

- **Monash Time Series Repository**: Benchmarks across many domains.
- **Dunnhumby Complete Journey**: Realistic retail data with seasonal dynamics.

```
# ADF Test (Original Series)
adf_original = adfuller(series)
print(f"ADF Statistic (original): {adf_original[0]:.4f}")
print(f"p-value (original): {adf_original[1]:.4f}")

# Differenced series
diff_series = series.diff().dropna()

# ADF Test (Differenced Series)
adf_diff = adfuller(diff_series)
print(f"ADF Statistic (differenced): {adf_diff[0]:.4f}")
print(f"p-value (differenced): {adf_diff[1]:.4f}")

# Fit models
h = 12

manual_model = ARIMA(order=(1, 1, 1), seasonal_order=(1, 1, 1, 12))
sf_manual = StatsForecast(models=[manual_model], freq='M')
sf_manual.fit(df)

auto_model = AutoARIMA(season_length=12)
sf_auto = StatsForecast(models=[auto_model], freq='M')
sf_auto.fit(df)

# Forecast
y_pred_manual = sf_manual.predict(h=h)
y_pred_auto = sf_auto.predict(h=h)

# Evaluation
truth = df.tail(h)['y'].values
print("MAE (Manual):", mean_absolute_error(truth, y_pred_manual['AirPassengers'].values))
print("MAE (Auto):", mean_absolute_error(truth, y_pred_auto['AirPassengers'].values))

# Plot forecasts
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['ds'], y=df['y'], mode='lines', name='Observed'))
fig.add_trace(go.Scatter(x=y_pred_manual['ds'], y=y_pred_manual['AirPassengers'], mode='l
fig.add_trace(go.Scatter(x=y_pred_auto['ds'], y=y_pred_auto['AirPassengers'], mode='lines

fig.update_layout(title='Manual SARIMA vs AutoARIMA Forecast',
                  xaxis_title='Date', yaxis_title='Passengers')
fig.show()
```

A.9 Practical Guide to ARIMA Model Selection

Choosing an appropriate ARIMA model is a highly iterative and diagnostic-driven process. While automatic procedures like AutoARIMA exist, a firm grasp of classical identification methods enables better judgment, more interpretable models, and greater robust-

ness—especially in edge cases. This guide synthesizes best practices, rules of thumb, and structural knowledge from Robert Nau and the Box-Jenkins methodology to support effective model selection in applied settings.

A.9.1 The Iterative ARIMA Model Selection Workflow

The ARIMA modeling process can be structured into five conceptual stages:

1. **Initial Data Inspection and Transformation**
   Examine the time series visually for trends, level shifts, seasonality, or variance instability. Apply transformations such as logging or inflation-adjustment as needed.
2. **Achieve Stationarity**
   Use differencing (nonseasonal and seasonal) to eliminate stochastic trends. Check for unit roots (e.g., via the Augmented Dickey-Fuller test). Aim to make the mean and variance constant over time.
3. **Identify AR and MA Orders**
   Analyze ACF and PACF plots of the differenced series:
   - ACF that cuts off and PACF that decays -> MA model (set q)
   - PACF that cuts off and ACF that decays -> AR model (set p)
   - Spikes in both ACF and PACF -> possible underdifferencing
4. **Estimate and Diagnose**
   Fit the model and check:
   - Residuals should resemble white noise (uncorrelated, constant variance)
   - Highest-order coefficients should be statistically significant (e.g., $|t| > 2$)
   - Residual ACF/PACF spikes may suggest increasing p or q
5. **Compare and Refine**
   Evaluate competing models with AIC/BIC. Prefer simpler models with well-behaved residuals. Iterate through previous steps if necessary.

Guiding Principles:

- **Parsimony:** Use the simplest model that fits the data well.
- **White Noise Residuals:** This is the ultimate diagnostic goal.
- **Iterate as Needed:** You may need to revisit earlier steps as new insights emerge.

Model Type	**Common Characteristics / Use Cases**	**Typical ARIMA Form**
Random Walk	Strong trend, no mean reversion; use for price series or cumulative metrics	ARIMA(0,1,0)
Random Walk with Drift	Trending behavior with constant average increment	ARIMA(0,1,0)+c
Mean-Reverting AR	Stationary process with memory and restoring force	ARIMA(p,0,0)+c
Shock-Absorbing MA	Stationary process driven by recent shocks, short memory	ARIMA(0,0,q)
Random Trend	Second-order nonstationarity; useful for stochastic trends	ARIMA(0,2,0)
Mixed Dynamics (ARMA)	Gradual decay in both ACF and PACF; often overparameterized unless clearly justified	ARIMA(p,0,q)
Common Seasonal ARIMA	Seasonal differencing and short memory patterns	ARIMA(0,1,1)$(0,1,1)_m$
Seasonal AR Model with Trend	Seasonally differenced trend + autoregressive structure	ARIMA(p,0,0)$(0,1,1)_m$ + c
AutoARIMA	Automated model search based on AIC/BIC; good for large-scale forecasting	Data-driven ARIMA(p,d,q)$(P,D,Q)_m$

Table A.2: Common ARIMA Models and Their Use Cases

A.9.2 Typical ARIMA Signatures and Use Cases

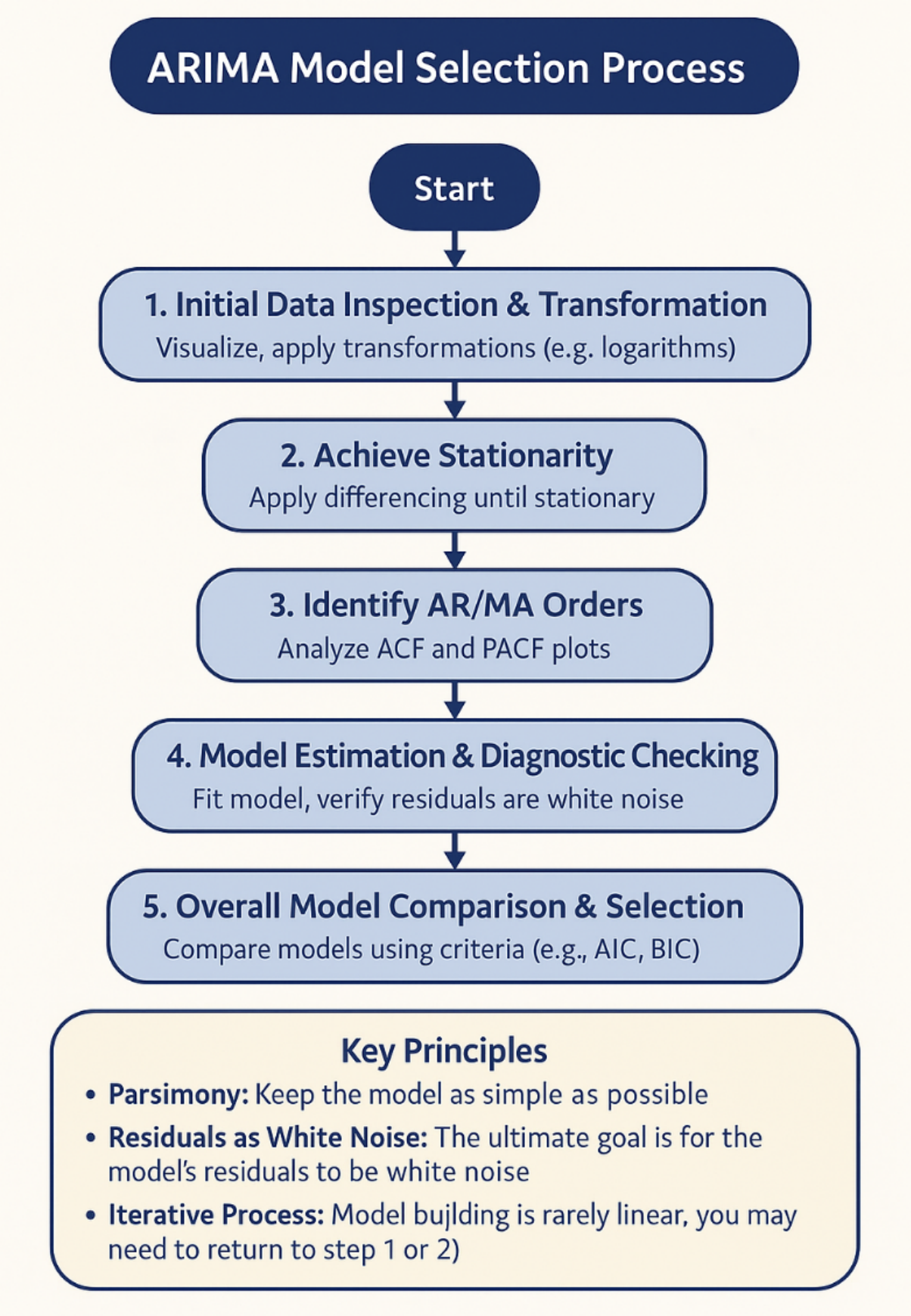

Figure A.8: A structured five-stage flowchart for selecting ARIMA models in time series forecasting. It begins with data inspection and transformation, followed by achieving stationarity, identifying AR/MA orders using ACF and PACF plots, estimating and diagnosing the model, and finally comparing models using information criteria like AIC or BIC. Emphasized guiding principles include model parsimony, ensuring residuals resemble white noise, and embracing the iterative nature of the modeling process.

A.10 Chapter Summary and Conclusions

In this chapter, we developed a comprehensive understanding of ARIMA modeling, covering both its theoretical foundation and practical application. Beginning with an overview of ARIMA's historical development and place within the broader family of time series models, we explored how autoregression (AR), integration (I), and moving average (MA) components combine to model various types of nonstationary data.

Key topics included:

- The role of differencing in achieving stationarity and the meaning of the "integrated" component.
- Interpreting ARIMA(p, d, q) models through ACF and PACF patterns.
- Rules of thumb and diagnostics for selecting appropriate AR and MA orders.
- Diagnostic checking of residuals for model validation, including coefficient significance and residual autocorrelation.
- Understanding unit roots and the implications of overdifferencing or inappropriate model complexity.
- Common ARIMA and SARIMA model forms, including their typical use cases in practice.
- A high-level model selection workflow designed to guide iterative model building and refinement.

Throughout, we emphasized core modeling principles: **parsimony**, the goal of **white noise residuals**, and the necessity of an **iterative model selection process**.

Looking Ahead: In the next chapter, we turn to *Exponential Smoothing* models—an alternative forecasting approach that models level, trend, and seasonality directly rather than through differencing and lag relationships. We will explore how ETS (Error-Trend-Seasonal) models relate to ARIMA, when to prefer them, and how to use them effectively in forecasting tasks.